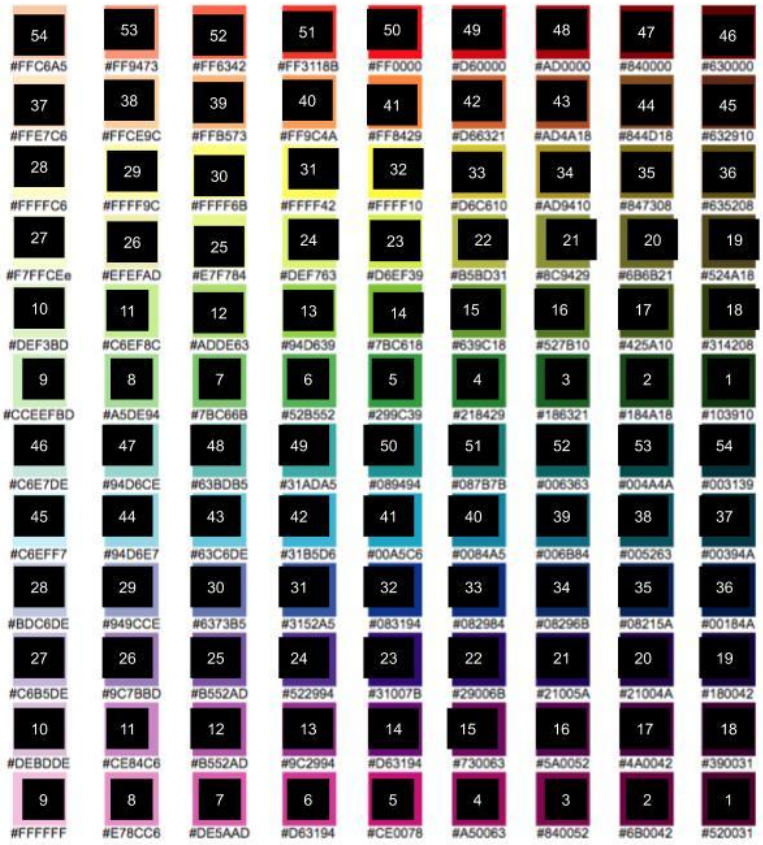
Le Match des couleurs

Le Match des couleurs is a flash and web-based piece composed of four html files. When exploring the piece through the Tate's Intermedia Art portal, the user first encounters the intro page:http://www2.tate.org.uk/netart/match/intro.htm. This intro page contains html, css, javascript and a gif. It acts as a gateway to either the html page test.htm, which contains a flash animation, or the frameset.htm page, which has the subframes test.html and dummy.html.

Inspection of the code shows that the piece is using the 1999 html standard declared at http://www.w3.org/1999/xhtml. Style elements of the page are declared in an embedded css file on lines 151-153 and inside tags as attributes. The html uses table tags to construct a grid of colors with 9 columns and 12 rows. Underneath each cell, there is a font tag that sets the font to Arial with Helvetica as a backup. This tag displays the hex code of the sub tables background color. Each cell in the table contains a subtable. The subtable contains an <a> tag which calls the file's javascript function R with a hardcoded number. The R function sets the value *document.source.clicked.value*=hval; and then runs the openIndex() function. The *document.source.clicked.value* is updated within a hidden form at the bottom of the page that tracks what was last clicked. The form's default value is 15 and it doesn't seem to be referenced anywhere after setting the value. This means that clicking any of the subtables has the same result.

The entries in the table have their numbers assigned in a slightly unorthodox manner, which I've illustrated in the photo below.



It's also notable that the intro.html file contains 22 js files. These files seem to have been added for the purpose of analytics. This is a list of the js files referenced:

- Analytics.js
- Www-widgetapi.js

- Main.2424edb5.js
- rules-p-yJp-XzbYwjHtL.js
- Scarab-v2.js
- Uwt.js
- Fbevents.js
- Aquant.js
- Hotjar-1586264.js
- Bat.js
- Core.js
- aMx.js
- Gtm.js
- Ga.js
- increasingly_tA7T9.js
- Modules.2500c3178bec9a890edb.js
- queueclientConfig.js
- Queueclient.min.js
- Queueconfigloader.min.js
- core.js
- ebAttribution.js
- tA7T9.js

References to these files occur in line 8, and from line 646-678, which means that a large chunk of the bottom of the file was likely edited at some point although there's no indication of when. Checking the metadata of the original html file may give a better idea of when this edit occurred.

Once the user has clicked on any of the squares described above, a new window will open. The openIndex() js function in the intro page calls a supporting js function, Is(), on line 25 to detect what type of machine the user has. This function affects which of the two files, frameset.htm or test.htm the file opens. The function then determines the height and width of the new window, depending upon whether the user has a Mac or a PC. Both cases have a conditional for if the screen height is less than 768 pixels. Therefore, we can be certain the artist wanted the window to be bigger than that height.

Both frameset.htm and test.htm have embed tags that specify that the height and width of the flash file should be 100%, indicating that the flash file should take up the entirety of the new window. However currently when viewed in modern browsers the embedded Flash file does not have these dimensions. Removing the explicit HTML declaration at the top of the files, <!DOCTYPE HTML PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd"> results in the embed tag being properly rendered as fullscreen with a width and height of 100%. The documentation for the XHTML 1.0 standard can be found here and explicitly states that it is equivalent to the HTML 4 Transitional standard.

According to this documentation from Mozilla and this documentation from W3 the embed tag was not a valid tag in earlier versions of HTML and only became valid in HTML5. See this stack overflow question for some embed tag history. This explains why it doesn't render full screen today, the standard it's using (XHTML 1.0), doesn't recognize embed as a valid tag. Removing that standard and letting the browser render it using newer HTML5 allows it to see the embed tag as valid and apply its attributes correctly.

This raises the question, did it ever display the embedded element full screen using the XHTML-1 Transitional format? Browsers change the way they parse formats and enforce strictness over time so it's possible that as HTML5 was adopted the browsers changed the way they enforced different HTML standard declarations and started enforcing rules they hadn't previously. This discrepancy will need to be addressed if a restoration is ever undertaken either by updating the file format or viewing it through an emulator that would correctly render the XHTML-1.0 Transitional format.

I don't think it's likely that the discrepancy with the size of the embed tag is due to a missing css file for two reasons. One there is still an embedded CSS file in the intro.html page, so at least one CSS file survived edits to the page. Two though the test.htm page seems to have had scripts edited and it's possible that something may have been removed during that process there is no CSS in the frameset.htm page which had no ads added to it and is designed to function almost identically to the test.htm page.

On line 88 under Mac conditions, the opener opens the test.html page with the window_width, window_height, window_top and window_left variables being passed to the opening function.
Under the conditions for if a user is using a pc on line 129 we see a commented out line:

```
//daddysLittleBoy=open("test3.htm","fullWin","fullscreen=1");
```

This line indicates that the artist either considered using an additional page at some point or renamed the file a few times and ultimately decided against this variable name. Line 131 opens the new window for the frameset.htm page with the window_width, window_height, window_top and window_left variables having been sent earlier based on the user's machine. Directly below on line 133, we see another commented out line for a simple opener with no page to open declared.

The two html files being opened ,test.htm for a mac user, and frameset.html for a pc user, are functionally very similar. Frameset.html contains a frameset with the dummy.html page on top and the test.htm page below. The dummy.html page has no content and seems to be a space filler. The title of the page is set to -. Notably, there is no declaration of an explicit html type in the frameset html page and no analytics have been added to this page. It's possible the declaration was overlooked when analytics were being added or it was created sometime after the other pages were edited.

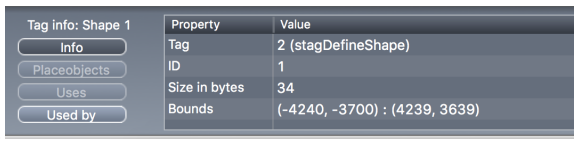
Test.html is also a fairly simple page. It uses the same 1999 html standard as the intro page and sets the page's title as -. Lines 7 to 27 contain commented out code that is intended to handle some Internet Explorer specific actions. Line 29 loads a js file intended to support running flash videos and lines 37 to 45 load the actual flash script. There is a noscript tag in case the browser doesn't load the initial embed tag.

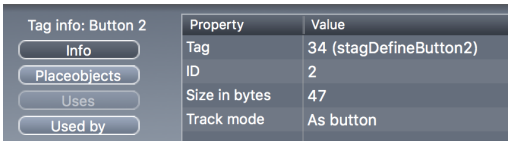
Test.html contains the following scripts, which all seem to relate to analytics:

- tA7T9.js
- Www-widgetapi.js
- rules-p-yJp-XzbYwjHtL.js
- Scarab-v2.js
- Uwt.js
- Fbevents.js
- Aquant.js
- Hotjar-1586264.js
- Bat.js
- https://s.pinimg.com/ct/core.js
- aMx.js
- Gtm.js
- Ga.js
- AC_RunActiveContent.js
- increasingly_tA7T9.js
- Modules.2500c3178bec9a890edb.js
- queueclientConfig.js
- Queueclient.min.js
- Queueconfigloader.min.js
- ebAttribution.js

Which brings us to the flash file. The flash file test.swf consists of 6496 frames and 86 sound files. The animation cycles through the same progression of colors and sound files with the colors gradually fading into each other. The file was decompiled using the software Trillix which allowed a deeper level of component analysis. The overall metadata of the file shows that it was made using SWF version 4. It has a width of 263 pixels and a height of 238 pixels, and the frame rate is 12.00 with a total of 6497 frames.

Trillix allows the user to see the individual components of a flash file. In this case, it revealed that the file used one shape, one button and two scripts. The shape and button metadata are shown in the screenshots below along with the two scripts' code:



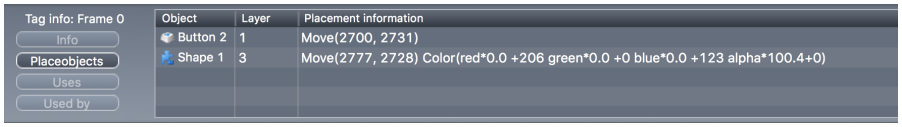


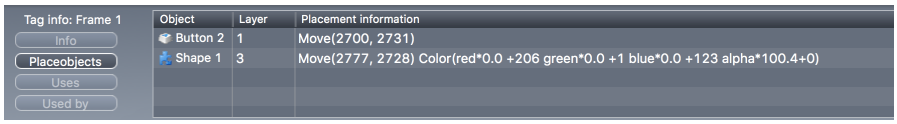

```
//Button 2
//  On press
on (press)
{
    getURL("FSCommand:exitme", "goaway");
}
```

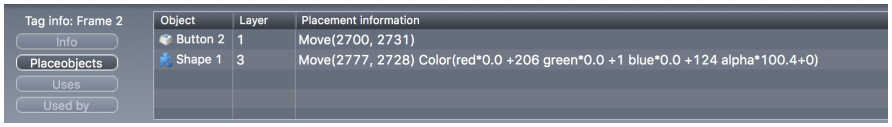
```
//Frame 6496
//  Action tag #0

gotoAndPlay(1);
```

The button is used by all of the frames but does not seem to be visible, and thus the script associated with it can be largely ignored. The second script is attached to the last frame in the sequence, frame 6496, and restarts the file on a loop.

When examining the individual frames of a flash file using Trillix, you are able to see the placement and color of the button and shape objects. The following screenshots show metadata on the first three files. For the first two frames, the color of the shape remains the same, while it changes slightly with the third frame. This shows that the gradual color shifting is hard coded and its order unchanging:







Further analysis could focus on recording this metadata information for all of the frames.

The final important piece of information that Trillix illuminates is which frames are associated with the sound files. The following table shows the name of the sound file and the frame that it appears in:

Sound file name	**Frame it appears in**
Sound 3	Frame 15
Sound 4	Frame 62

Sound 5	Frame 124
Sound 6	Frame 204
Sound 7	Frame 279
Sound 8	Frame 337
Sound 9	Frame 393
Sound 10	Frame 455
Sound 11	Frame 562
Sound 12	Frame 641
Sound 13	Frame 699
Sound 14	Frame 782
Sound 15	Frame 857
Sound 16	Frame 924
Sound 17	Frame 997
Sound 18	Frame 1069
Sound 19	Frame 1118
Sound 20	Frame 1183
Sound 21	Frame 1257
Sound 22	Frame 1311
Sound 23	Frame 1397
Sound 24	Frame 1475
Sound 25	Frame 1557
Sound 26	Frame 1639
Sound 27	Frame 1697
Sound 28	Frame 1760
Sound 29	Frame 1818
Sound 30	Frame 1904
Sound 31	Frame 1965
Sound 32	Frame 2040
Sound 33	Frame 2103
Sound 34	Frame 2158
Sound 35	Frame 2253

Sound 36	Frame 2332
Sound 37	Frame 2388
Sound 38	Frame 2446
Sound 39	Frame 2532
Sound 40	Frame 2610
Sound 41	Frame 2671
Sound 42	Frame 2753
Sound 43	Frame 2840
Sound 44	Frame 2902
Sound 45	Frame 2988
Sound 46	Frame 3079
Sound 47	Frame 3133
Sound 48	Frame 3218
Sound 49	Frame 3302
Sound 50	Frame 3386
Sound 51	Frame 3474
Sound 52	Frame 3565
Sound 53	Frame 3646
Sound 54	Frame 3729
Sound 55	Frame 3829
Sound 56	Frame 3876
Sound 57	Frame 3953
Sound 58	Frame 4010
Sound 59	Frame 4093
Sound 60	Frame 4187
Sound 61	Frame 4233
Sound 62	Frame 4284
Sound 63	Frame 4370
Sound 64	Frame 4431
Sound 65	Frame 4482
Sound 66	Frame 4568

Sound 67	Frame 4655
Sound 68	Frame 4736
Sound 69	Frame 4791
Sound 70	Frame 4848
Sound 71	Frame 4932
Sound 72	Frame 4985
Sound 73	Frame 5069
Sound 74	Frame 5166
Sound 75	Frame 5251
Sound 76	Frame 5336
Sound 77	Frame 5416
Sound 78	Frame 5504
Sound 79	Frame 5597
Sound 80	Frame 5698
Sound 81	Frame 5793
Sound 82	Frame 5896
Sound 83	Frame 5989
Sound 84	Frame 6078
Sound 85	Frame 6172
Sound 86	Frame 6238
Sound 87	Frame 6328
Sound 88	Frame 6383
Sound 89	Frame 6470