headers

#include <gdk-pixbuf/gdk-pixbuf.h>

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

GdkPixbuf\* logo\_pixbuf = gdk\_pixbuf\_new\_from\_file("/home/rakesh/Downloads/logo.png", NULL);

if (!logo\_pixbuf) {

g\_print("Error loading the logo image.\n");

return 0;

}

gint logo\_width = gdk\_pixbuf\_get\_width(logo\_pixbuf);

gint logo\_height = gdk\_pixbuf\_get\_height(logo\_pixbuf);

g\_print("height=%d width=%d\n", logo\_height, logo\_width);

guint8\* logo\_data = gdk\_pixbuf\_get\_pixels(logo\_pixbuf);

//\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

GdkPixbuf\* scaled\_logo\_pixbuf = gdk\_pixbuf\_scale\_simple(logo\_pixbuf, new\_width, new\_height, GDK\_INTERP\_BILINEAR);

guchar\* scaled\_logo\_data;

int rowstride;

if (gdk\_pixbuf\_read\_pixels(scaled\_logo\_pixbuf, &scaled\_logo\_data, &rowstride, 0, 0)) {

// You now have access to the pixel data in scaled\_logo\_data.

// The rowstride indicates the number of bytes in each row of the image.

} else {

g\_print("Error reading pixel data from scaled logo image.\n");

// Handle the error

}

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

int width = gdk\_pixbuf\_get\_width(scaled\_logo\_pixbuf);

int height = gdk\_pixbuf\_get\_height(scaled\_logo\_pixbuf);

for (int y = 0; y < height; y++) {

for (int x = 0; x < width; x++) {

// Access pixel at (x, y)

guchar\* pixel = scaled\_logo\_data + y \* rowstride + x \* 3; // Assuming 3 bytes per pixel for RGB data

// Now 'pixel' points to the RGB color of the pixel at (x, y).

// You can access the individual color channels (red, green, blue) using 'pixel[0]', 'pixel[1]', and 'pixel[2]'.

}

}

//------------------------------------------------------------------------------------------------

// Assuming 'pixel' contains RGB values in the format R, G, B.

guchar r = pixel[0]; // Red channel

guchar g = pixel[1]; // Green channel

guchar b = pixel[2]; // Blue channel

// Calculate Y (luma) value.

guchar y = (66 \* r + 129 \* g + 25 \* b + 128) >> 8 + 16;

// Calculate U (chroma) and V (chroma) values. NV12 format uses the same U and V for 2x2 pixel blocks.

guchar u = (-38 \* r - 74 \* g + 112 \* b + 128) >> 8 + 128;

guchar v = (112 \* r - 94 \* g - 18 \* b + 128) >> 8 + 128;

//\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

int width = gdk\_pixbuf\_get\_width(logo\_pixbuf);

int height = gdk\_pixbuf\_get\_height(logo\_pixbuf);

int channels = gdk\_pixbuf\_get\_n\_channels(logo\_pixbuf); // This will give you the number of color channels (e.g., 3 for RGB, 4 for RGBA).

guint8\* pixel\_data = gdk\_pixbuf\_get\_pixels(logo\_pixbuf);

for (int y = 0; y < height; y++) {

for (int x = 0; x < width; x++) {

int offset = y \* gdk\_pixbuf\_get\_rowstride(logo\_pixbuf) + x \* channels;

guint8 red = pixel\_data[offset];

guint8 green = pixel\_data[offset + 1];

guint8 blue = pixel\_data[offset + 2];

g\_print("Pixel at (%d, %d): Red=%d, Green=%d, Blue=%d\n", x, y, red, green, blue);

}

}

<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<------------------->>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>

#include <stdio.h>

#include <gdk-pixbuf/gdk-pixbuf.h>

int main() {

// Load the PNG image into a GdkPixbuf

GdkPixbuf\* logo\_pixbuf = gdk\_pixbuf\_new\_from\_file("logo.png", NULL);

if (!logo\_pixbuf) {

g\_print("Error loading the logo image.\n");

return 1;

}

// Assuming NV12 video frame parameters

int video\_frame\_width = 1920;

int video\_frame\_height = 1080;

// Scale the GdkPixbuf to match the video frame dimensions

GdkPixbuf\* scaled\_logo\_pixbuf = gdk\_pixbuf\_scale\_simple(logo\_pixbuf, video\_frame\_width, video\_frame\_height, GDK\_INTERP\_BILINEAR);

int width = gdk\_pixbuf\_get\_width(scaled\_logo\_pixbuf);

int height = gdk\_pixbuf\_get\_height(scaled\_logo\_pixbuf);

// Create a buffer to store the NV12 image

int nv12\_size = video\_frame\_width \* video\_frame\_height \* 3 / 2;

guint8\* nv12\_data = g\_malloc(nv12\_size);

guint8\* rgb\_data = gdk\_pixbuf\_get\_pixels(scaled\_logo\_pixbuf);

for (int y = 0; y < height; y++) {

for (int x = 0; x < width; x++) {

int rgb\_offset = (y \* width + x) \* 3;

guint8 r = rgb\_data[rgb\_offset];

guint8 g = rgb\_data[rgb\_offset + 1];

guint8 b = rgb\_data[rgb\_offset + 2];

// Convert RGB to YUV (NV12)

guint8 y = (66 \* r + 129 \* g + 25 \* b + 128) >> 8 + 16;

guint8 u = (-38 \* r - 74 \* g + 112 \* b + 128) >> 8 + 128;

guint8 v = (112 \* r - 94 \* g - 18 \* b + 128) >> 8 + 128;

int nv12\_y\_offset = y \* video\_frame\_width + x;

int nv12\_uv\_offset = video\_frame\_width \* video\_frame\_height + (y / 2) \* (video\_frame\_width / 2) + (x / 2);

nv12\_data[nv12\_y\_offset] = y;

nv12\_data[nv12\_uv\_offset] = u;

nv12\_data[nv12\_uv\_offset + 1] = v;

}

}

// Now you have the NV12 data in the 'nv12\_data' buffer. You can use this data as needed.

// Don't forget to free the memory when done

g\_free(nv12\_data);

return 0;

}