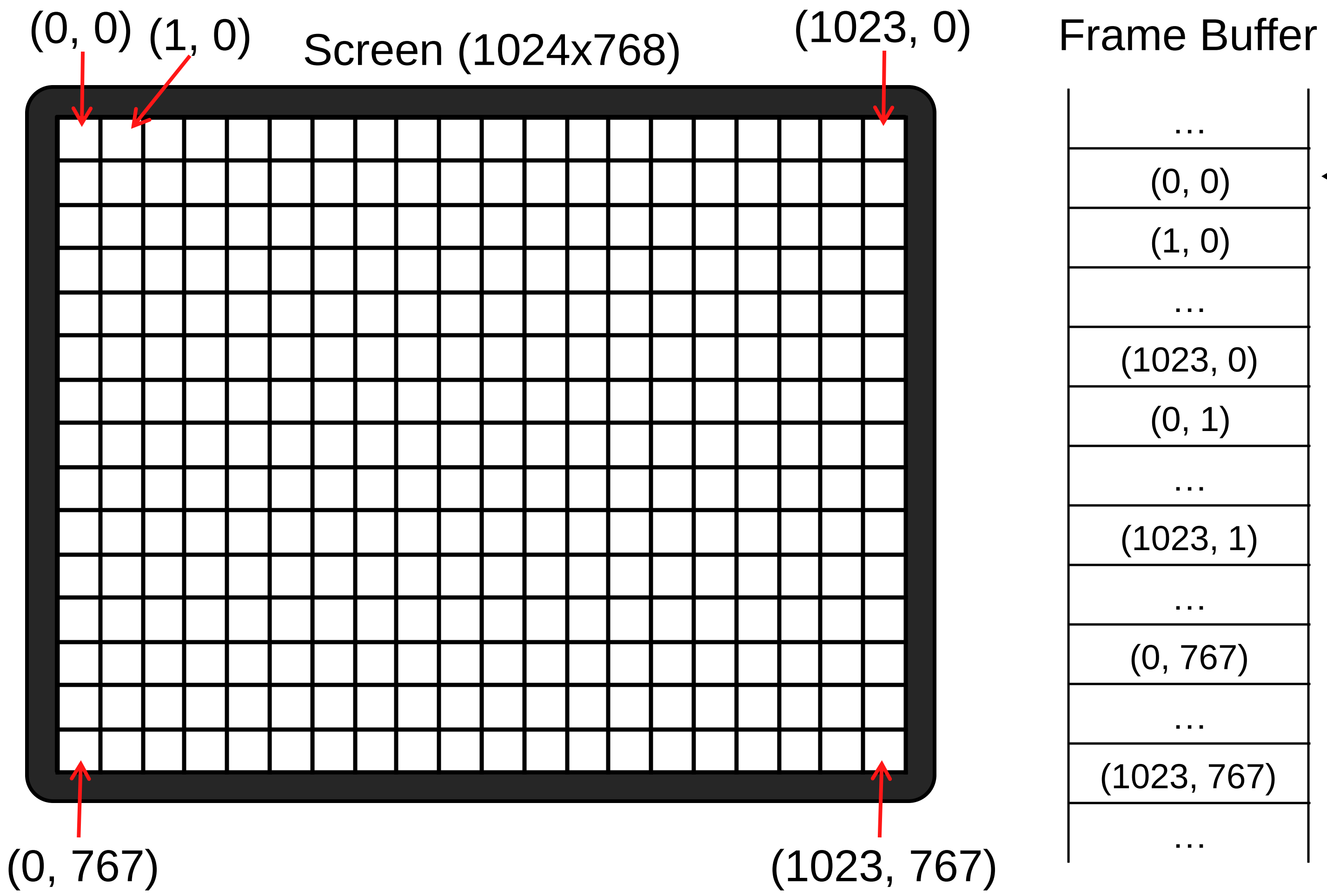
# CPSC 359 – Tutorial #7

## Video Interface

Modified from Andrew Kuipers

Updated for RPi2 / Spring 2016

# Video Interface



# Video Interface

- Draw pixels by writing colour values to the Frame Buffer

  addr(x, y) = base pointer + ((y * width) + x) * (bpp / 8)

- Colour value is split into Red, Green and Blue colour channels
  - Higher values in a channel mean more of that colour

| | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **Low Colour Mode (8bpp)** | **7** | **6** | **5** | **4** | **3** | **2** | **1** | **0** | | | | |
| **High Colour Mode (16bpp)** | **15** | **…** | **11** | **10** | **…** | **5** | **4** | **…** | **0** | | | |
| **True Colour Mode (24bpp)** | **23** | **…** | **16** | **15** | **…** | **8** | **7** | **…** | **0** | | | |
| **RGBA32 Mode (32bpp)** | **31** | **…** | **24** | **23** | **…** | **16** | **15** | **…** | **8** | **7** | **…** | **0** |

## Video Interface

- Before we can draw pixels, we need to:
  1. Set the Resolution (width & height in pixels) of the display
  2. Set the Bit Depth (bits per pixel) of the display
  3. Get a pointer to the Frame Buffer
- Need to interface with the GPU to accomplish this
  - Raspberry Pi uses a Mailbox interface to talk to the GPU

# Video Interface

Initialize Frame Buffer via Mailbox Interface

1. Create a data structure containing initialization information
2. Wait until Mailbox can accept a message
3. Write address of init. struct  to Mailbox Frame Buffer Channel
4. Wait for response from Mailbox
5. Wait for Frame Buffer pointer in init. struct to be set

## Image Bitmap

- Check on D2L the “ImagetoASCII” Java application for converting an Image to ASCII bitmap structure.
- Save it in the `.data` section as ASCII structure.
- Create a function that loads 16-bits color values [***half-words***] and stores into the frame buffer.
- The ASCII bitmap structure created is a 1-D array that contains 16-bits color values in row-major order.
- Use it for picking 16-bits ***hex*** color code as well.

## Challenge

- Download a 16x16 pixels image.
- Convert using “ImagetoASCII”.
- Write a function to draw your image on the screen:
  - Arguments:
    - Address of the image data.
    - X & Y coordinate to place your image on the screen.