

Computer Graphics  
*Theory*

Christian Rossi

Academic Year 2023-2024

## **Abstract**

The course outline is:

- Rendering pipeline, and hardware and software architectures for 3D graphics.
- Basic transformation: translation, rotation, scaling and projection.
- Basic of computational geometry, clipping and hidden surface removal.
- Lighting: light sources, materials, shaders, surface normal.
- Texture: projection, mapping, texture animation, alpha mapping, bump mapping, normal mapping.
- Advanced effects: reflection maps, BRDF models, environment maps, global illumination maps.
- Animation: scene graph, Bézier curves, quaternion.

---

# Contents

---

<b>1</b>	<b>Introduction</b>	<b>1</b>
1.1	Graphic adapter . . . . .	1
1.2	Colors . . . . .	2
1.2.1	Physical phenomena . . . . .	2
1.2.2	Color reproduction . . . . .	2
1.3	Image resolution . . . . .	3
1.3.1	Coordinates . . . . .	3
1.3.2	Normalized screen coordinates . . . . .	4
1.3.3	Graphics primitives . . . . .	5

# CHAPTER 1

## Introduction

### 1.1 Graphic adapter

Graphics adapters partition the screen into a grid of discrete elements known as pixels, each capable of displaying a specific color. These pixels collectively render images sampled from spatial data, enabling the adapter to accurately display them on a monitor. To facilitate this process, the adapter is equipped with dedicated memory known as video memory (VRAM).

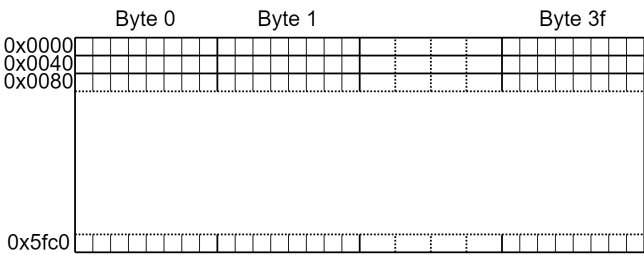


Figure 1.1: VRAM's structure

A segment of the VRAM, referred to as the screen buffer or frame buffer, contains encoded color information for all the pixels displayed on the screen. A specialized component on the graphics card, such as the RAMDAC for analog displays, utilizes this information from the VRAM to construct the image on the display. Users typically do not interact directly with the screen buffer; instead, they store images in various sections of the VRAM. Subsequently, users compose the on-screen image by issuing commands to the graphics adapter. These commands can include drawing points, lines, and other shapes, writing text, transferring raster images from the VRAM, performing 3D projections, and applying deformations and effects to the images. By combining these commands with the data stored in the VRAM, the adapter constructs the final image and transmits it to the display.

Performing complex operations involves various tasks such as interacting with multiple displays, managing multiple graphic adapters, or displaying multiple images simultaneously (e.g., stereoscopy), which can introduce significant complexity.

**Vulkan** Graphics adapters are typically developed in conjunction with their software drivers, ensuring that programmers interface with the hardware through libraries provided by the man-

ufacturer rather than directly accessing video card registers. Vulkan exemplifies a platform-independent standard set of procedures that streamlines the access of graphic card functionalities by application code.

## 1.2 Colors

Color coding is a fundamental aspect of computer graphics, dictating how colors displayed on-screen are represented as sequences of binary digits. Typically, on-screen colors are encoded using the RGB system, which stands for Red-Green-Blue.

### 1.2.1 Physical phenomena

In terms of physics, the color of light is determined by the wavelength of the photons it emits. When a light source emits photons of various wavelengths, objects interact with these photons based on their composition, either reflecting or absorbing them at different intensities. The wavelengths of reflected photons contribute to the primary colors of an object. These reflected photons are then focused by the lens of the human eye onto the retina, where specialized cells called rods and cones reside. Rods are sensitive to light intensity, while cones are sensitive to light color. Through these cells, the brain processes visual information. There are three distinct types of cones distributed evenly across the retina, each responsive to a specific portion of the light spectrum. The brain integrates signals from these cones to perceive a given color.

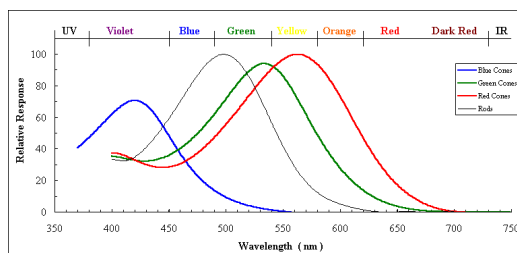


Figure 1.2: Human cones wavelength

### 1.2.2 Color reproduction

Color reproduction operates on the principle inverse to human vision, employing distinct emitters for each color perceptible by specific types of cones in the eye. As the wavelengths perceived by the three types of cones primarily correspond to red, green, and blue, different hues are created by blending the light from these three primary colors. When observing a display from a sufficient distance, the human eye naturally blends the primary hues, reproducing the stimuli associated with a combined color.

By mixing two of the three primary colors, such as red and green or blue and red, we produce secondary colors like yellow, cyan, or magenta. Mixing all three primaries yields white, while adjusting the proportions of the three colors enables the reproduction of various hues.

Comparing the wavelengths of photons sensed by cones with the frequency spectrum helps understand why cyan results from blending blue and green, and yellow emerges from mixing red and green. The visible spectrum to the human eye is broad, characterized by a parabolic shape.

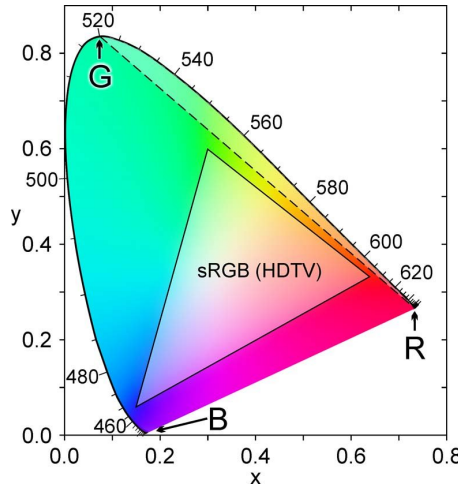


Figure 1.3: Colors perceived by human eyes and reproduced by a monitor

The range of colors that a monitor can display corresponds to a cube, with the red, green, and blue components positioned along the  $x$ ,  $y$  and  $z$ -axis, respectively.

The levels of these components translate into electrical signals that regulate the intensity of light emitted by the screen for each primary color. In digital systems, these signals are typically generated through DAC (Digital-to-Analog Converter) conversion, resulting in quantization, which reduces the number of available colors.

Different monitors may assign various frequencies of the spectrum to each primary color and encode their levels differently. Color profiles are used to compensate for these variations, ensuring consistent behavior across different adapters.

## 1.3 Image resolution

Image resolution refers to the pixel density within a given physical unit, typically measured in DPI (Dots Per Inch). In the context of raster graphic devices, resolution is relative to the size of the monitor rather than being an absolute metric as in printed images. Consequently, the resolution of a screen specifies the number of pixels displayed horizontally ( $s_w$ ) and vertically ( $s_h$ ). It's important to note that pixels may not have a square shape, and as a result, the horizontal resolution often differs from the vertical resolution.

### 1.3.1 Coordinates

The coordinate system utilized is Cartesian, with the origin situated in the top-left corner. The  $x$ -axis progresses horizontally from left to right, while the  $y$ -axis ascends vertically from top to bottom.

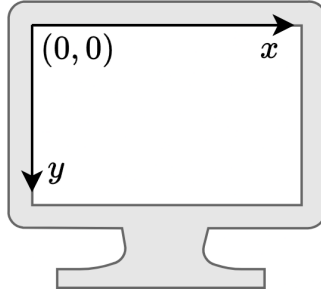


Figure 1.4: Pixel coordinates

The coordinate system adheres to a left-handed convention, signifying that the  $y$ -axis extends in the opposite direction compared to the conventional Cartesian system. As a consequence, the integer values of  $x$  and  $y$  coordinates fall within the range:

$$0 \leq x \leq s_w - 1$$

$$0 \leq y \leq s_h - 1$$

where  $s_w$  and  $s_h$  represent the horizontal and vertical dimensions of the screen, respectively. It's noteworthy that coordinates may exceed the boundaries of the screen.

**Definition** (*Clipping*). Clipping involves trimming the primitives to exhibit only their visible segments.

Failure to execute clipping may result in undesirable outcomes such as wrapping around or writing to unallocated memory space, often leading to irreparable errors depending on the hardware.

Contemporary displays offer diverse resolutions, sizes, and form factors. Applications strive to maintain consistent content presentation across varying resolutions, sizes, and screen shapes while harnessing the full potential of the display's capabilities.

### 1.3.2 Normalized screen coordinates

Normalized screen coordinates provide a standardized method for addressing points on a screen in a manner that is independent of the device's size and shape. As screens may possess varying aspect ratios and non-square pixels, the objective is to represent the same scene consistently, irrespective of the actual proportions or resolution, by adjusting or adding features based on available space. This concept extends to windows in conventional operating systems, which users can freely resize.

Many applications render images to memory, where the proportions of the display area can be arbitrary. Normalized screen coordinates utilize a Cartesian coordinate system, with  $x$  and  $y$  values ranging between two canonical values (commonly between -1 and 1, although other standards like 0 and 1 also exist), and axes oriented along specific directions.

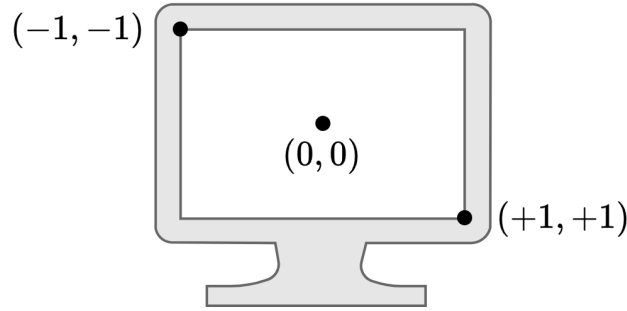


Figure 1.5: Normalized screen coordinates

For example, both OpenGL and Vulkan employ normalized screen coordinates within the  $[-1, 1]$  range. However, OpenGL's  $y$ -axis increases upwards, while Vulkan adheres to the convention of pixel coordinates with the  $y$ -axis descending downwards.

Given a screen (or window, or memory area) resolution of  $s_w \times s_h$  pixels. Pixel coordinates  $(x_s, y_s)$  can be derived from normalized screen coordinates  $(x_n, y_n)$  using a straightforward relation. For Vulkan, the transformation is expressed as:

$$\begin{cases} x_s = (s_w - 1) \cdot \frac{x_n + 1}{2} \\ y_s = (s_h - 1) \cdot \frac{y_n + 1}{2} \end{cases}$$

As mentioned, screen buffers are typically accessed through their drivers and specific software libraries. Programs generally utilize normalized screen coordinates, except in cases where they operate at a low level and must directly interact with the frame buffer.

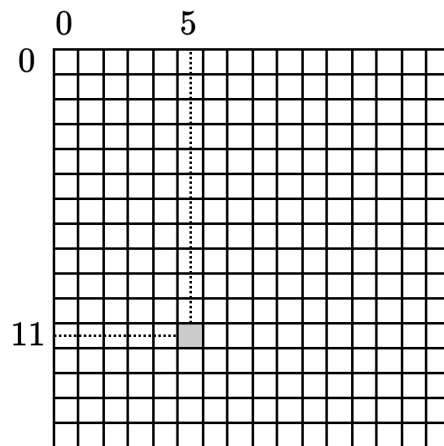
### 1.3.3 Graphics primitives

Procedures responsible for drawing simple geometric shapes on a screen, utilizing a 2D coordinate system, are known as 2D graphics primitives. Modern graphics adapters facilitate drawing operations by supporting three fundamental types of primitives: points, lines, and filled triangles.

These primitives work by manipulating and connecting points on the screen, identified by a pair of coordinates defined with a two-component vector. These coordinates are integer values measured in pixels. The primitives employed for screen drawing (or within windows) automatically handle the calculations to determine the appropriate pixels on the screen based on normalized screen coordinates.

**Point** Drawing a point entails setting the color of a pixel at a specified position on the screen. The graphic primitive responsible for this action is typically called `plot()`. It requires parameters such as the coordinates of the pixel to be set  $(x, y)$  and its color  $(r, g, b)$ .



Figure 1.6: `plot(x:-0.312, y:0.562, r:0.8, g:0.8, b:0.8)`

**Line** The line primitive connects two points on the screen with a straight segment. It necessitates the coordinates  $(x_0, y_0)$  of the starting point and  $(x_1, y_1)$  of the end point, along with the color definition.

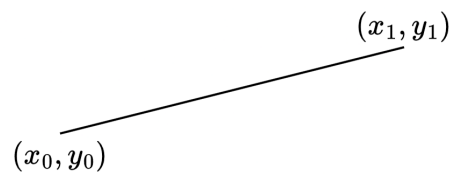


Figure 1.7: Line primitive

**Filled triangle** Filled triangles serve as the foundation of 3D computer graphics. They are defined by the coordinates of their three vertices  $(x_0, y_0)$ ,  $(x_1, y_1)$ , and  $(x_2, y_2)$ , along with their corresponding color  $(r, g, b)$ .

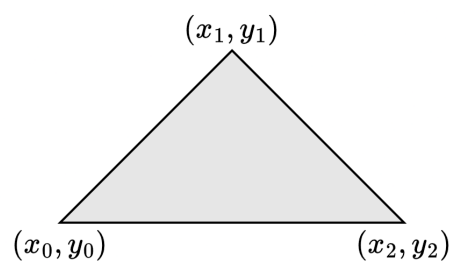


Figure 1.8: Filled triangle primitive