UNIVERSITY OF MINNESOTA
Driven to Discover®

Computer Graphics Basics

CSCI 4611: Programming Interactive Computer Graphics and Games

Evan Suma Rosenberg | CSCI 4611 | Fall 2022

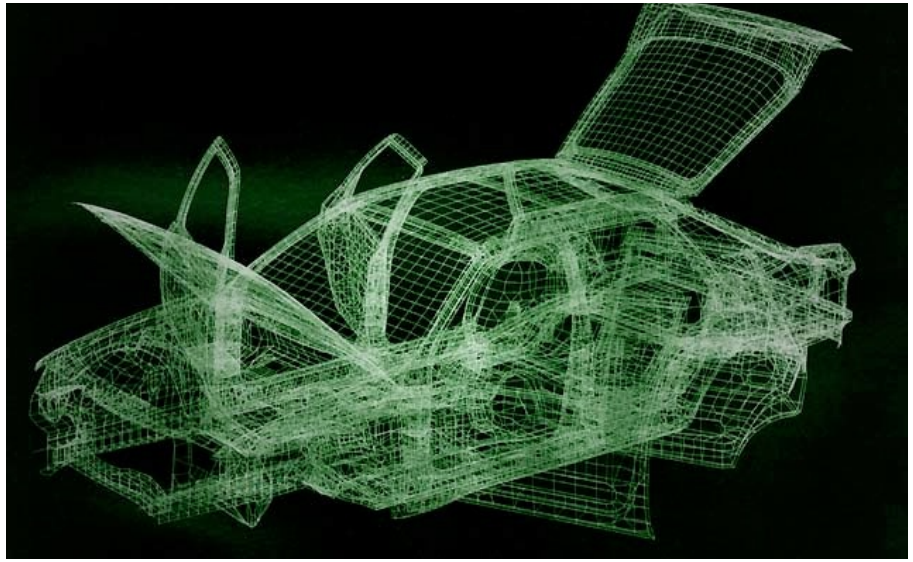
Computer Graphics Paradigms

Sample-Based Graphics

- Discrete samples are used to describe visual information
- Pixels can be created by digitizing images, using a sample-based "painting" program, etc.

  Often some aspect of the physical world is sampled for visualization

  Example programs: Adobe Photoshop™, GIMP™ , Adobe AfterEffects™

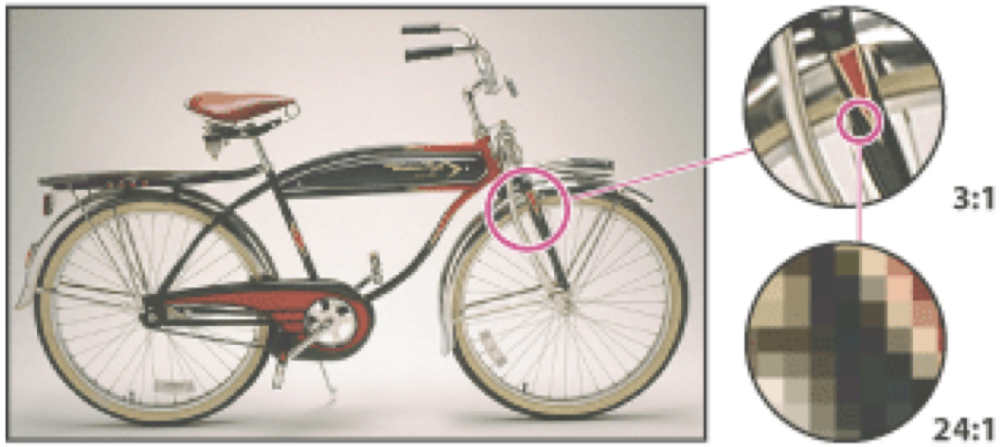


Geometry-Based (Scalable Vector) Graphics

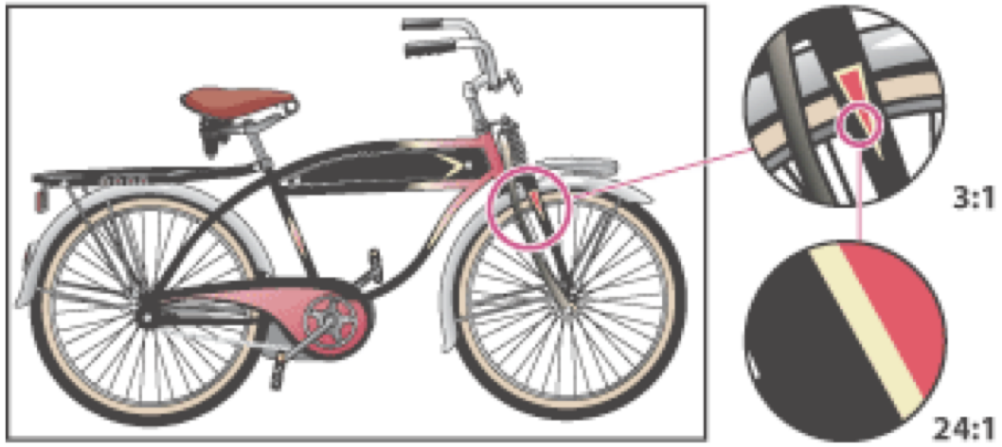
- Geometrical model is created, along with various appearance attributes, and is then sampled for visualization (rendering)
- Examples of 2D apps: Adobe Illustrator™, Adobe Freehand™ (formerly by Macromedia), Corel CorelDRAW™

  Examples of 3D apps: Autodesk's AutoCAD™, Autodesk's Maya™, Autodesk's 3D Studio Max™

Sample-Based vs. Geometry-Based Graphics



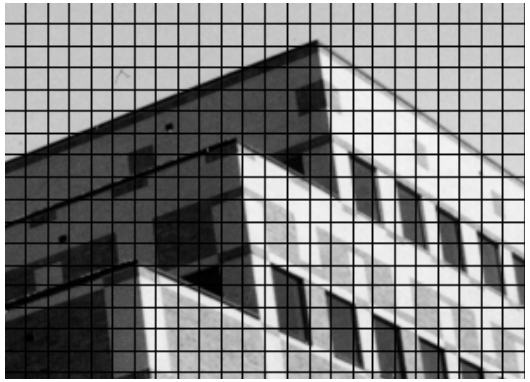
For sample-based graphics, the original data is itself a sample (like a digital photo) of the real world, so there's no way to get any finer resolution that the original sample when you zoom in.



With geometry-based graphics, the computer has an underlying geometric/mathematical representation of the object, so you can zoom in and redraw from a closer view.

Don't be confused — of course, any time you draw to a screen you are in a sense "sampling" because you have to get the picture to fit into pixels — the key with geometry-based graphics is that the underlying geometric model makes it possible to draw at any scale.

Sampling a Scene

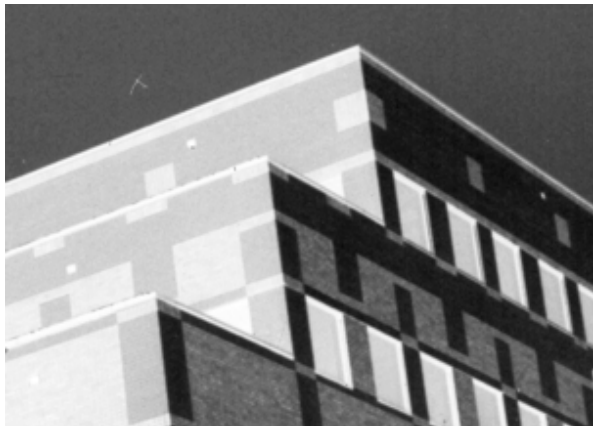


10 = white, 5 = gray, 0 = black

- Lets do some sampling of a building
- A color value is measured at every grid point and used to color corresponding grid square

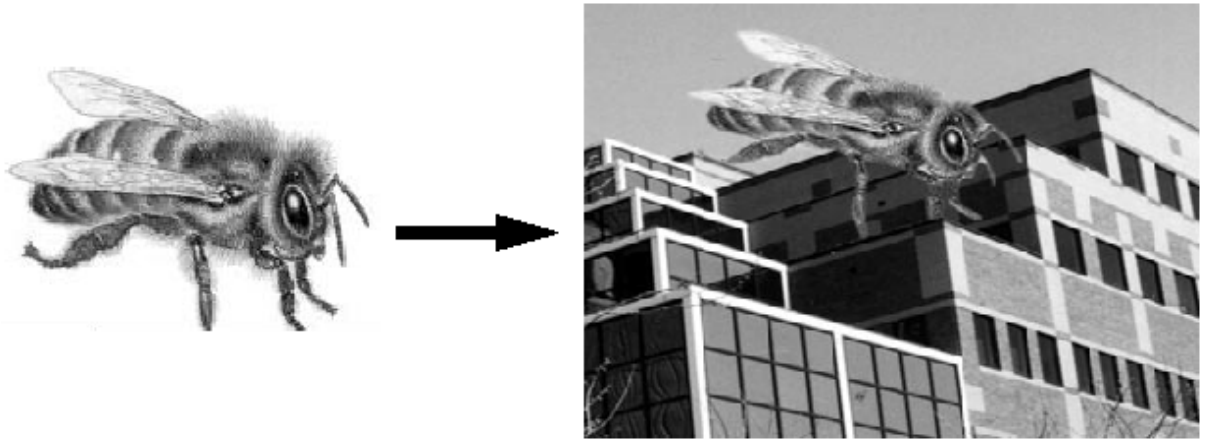
  This poor sampling and image reconstruction method creates a blocky image

Advantages of Sample-Based Graphics



Once image is defined in terms of colors at (x, y) locations on the grid, you can change the image easily by altering the location or color values.

For example, if we reverse our mapping and make 0 = white and 10 = black, the image would look like this.



Pixel information from one image can be copied and pasted into another, replacing or combining them with previously stored pixels.

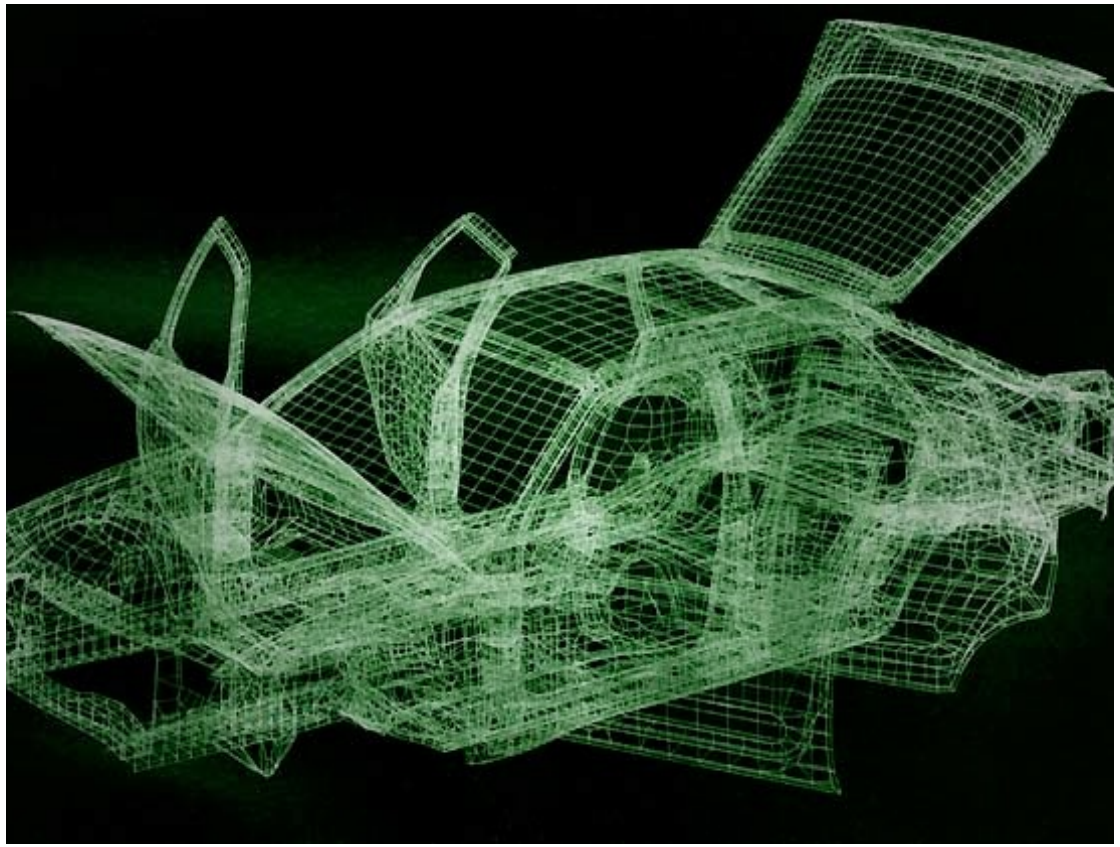
Disadvantages of Sample-Based Graphics

Photo Tourism: Exploring Photo Collections in 3D (SIGGRAPH 2006)

- WYSIAYG (What You See Is All You Get)
  - No additional information
  - no depth information
  - can’t examine scene from different point of view
  - at most can play with the individual pixels or groups of pixels to change colors, enhance contrast, find edges, etc.

Recently, there is strong interest in image-based rendering to fake 3D scenes and arbitrary camera positions. New images can be constructed by interpolation, composition, warping and other operations. (Check out our courses on computer vision!)

Geometry-Based Graphics



- Geometry-based graphics applications store mathematical descriptions, or "models," of geometric elements (lines, polygons, polyhedrons…) and associated attributes (e.g., color, material properties).
- Elements are primitive geometric shapes, primitives for short
- Images created as pixel arrays (via sampling of geometry) for viewing, but not stored as part of model. Images of many different views are generated from same model.
- Users cannot usually work directly with individual pixels in geometry-based programs; as user manipulates geometric elements, program resamples and redisplays elements

Combining Geometry and Sample-Based Graphics

3D model

(geometry-based graphics)

+

image texture

(sample-based graphics)

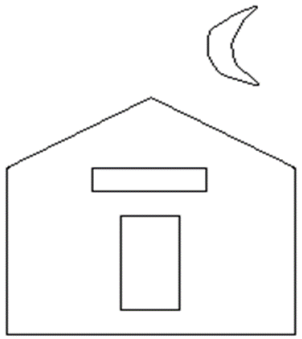
=

steve dabbing

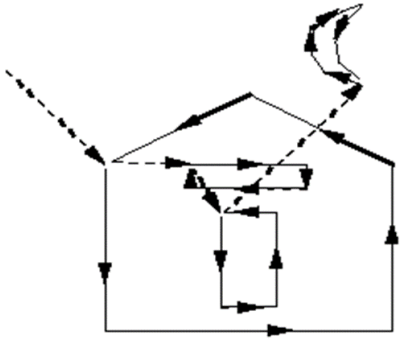
(combined)

Rendering often combines geometric and sample-based graphics, both as a performance hack and to increase the quality of the final product (e.g., we will combine "image textures" with 3D geometries).

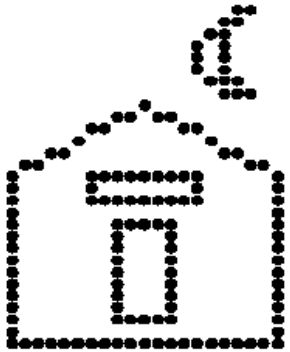
Drawing Graphics



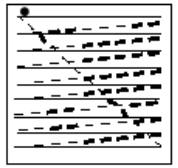
Ideal Drawing



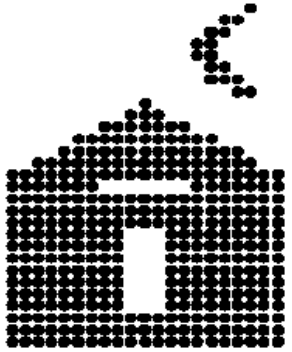
Vector Drawing



outline primitives



Raster



filled primitives

Vector Drawing

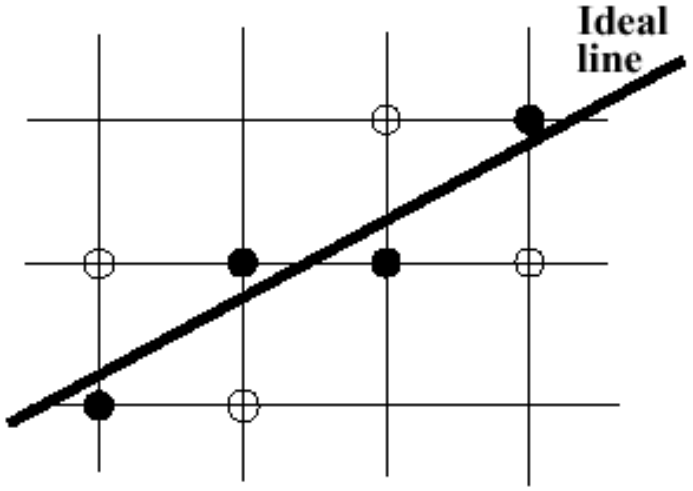
calligraphic, stroke, random-scan displays or plotters

Raster Drawing

monitors, TVs, smartphones, laser printers, etc.

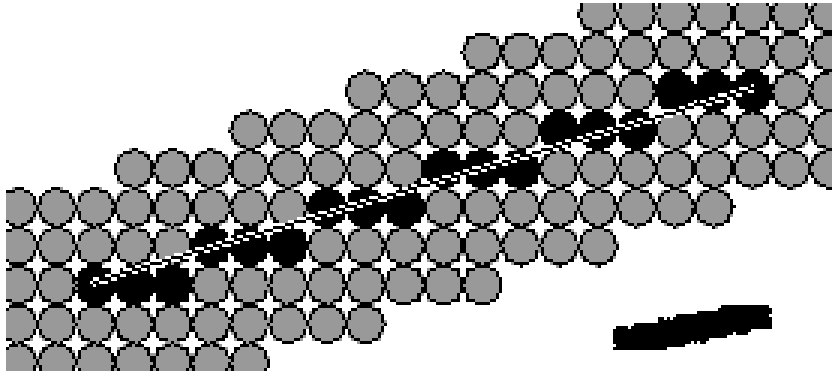
Drawing Lines

- For horizontal, vertical and diagonal lines where all pixels lie on ideal line: special case



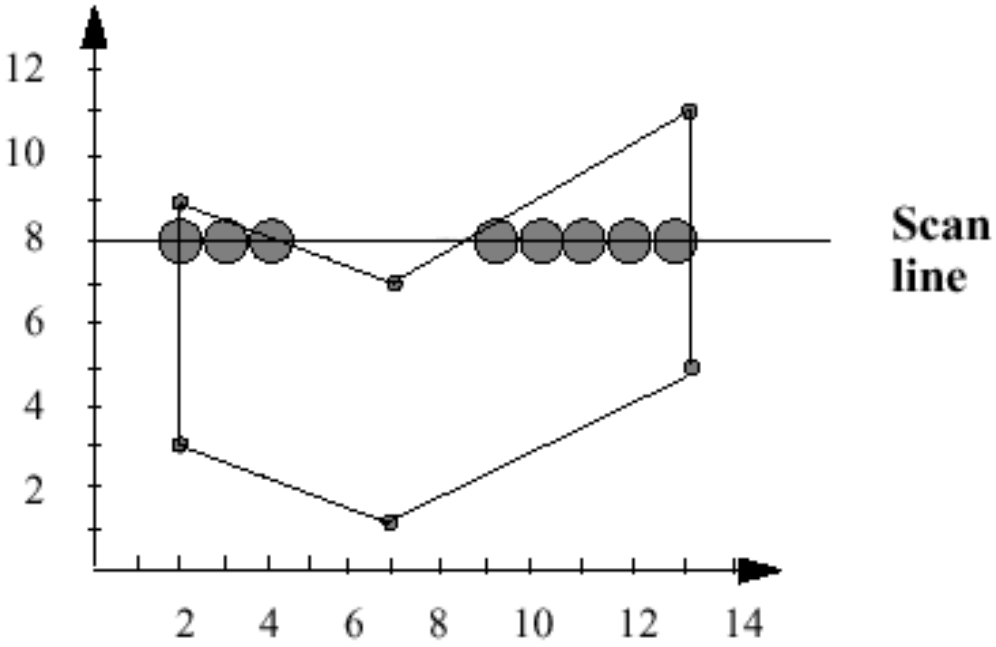
- For lines at an arbitrary angle, select pixels closest to the ideal line (Bresenham’s midpoint "scan conversion" algorithm)

- Sampling continuous line on discrete grid introduces sampling errors: "jaggies"

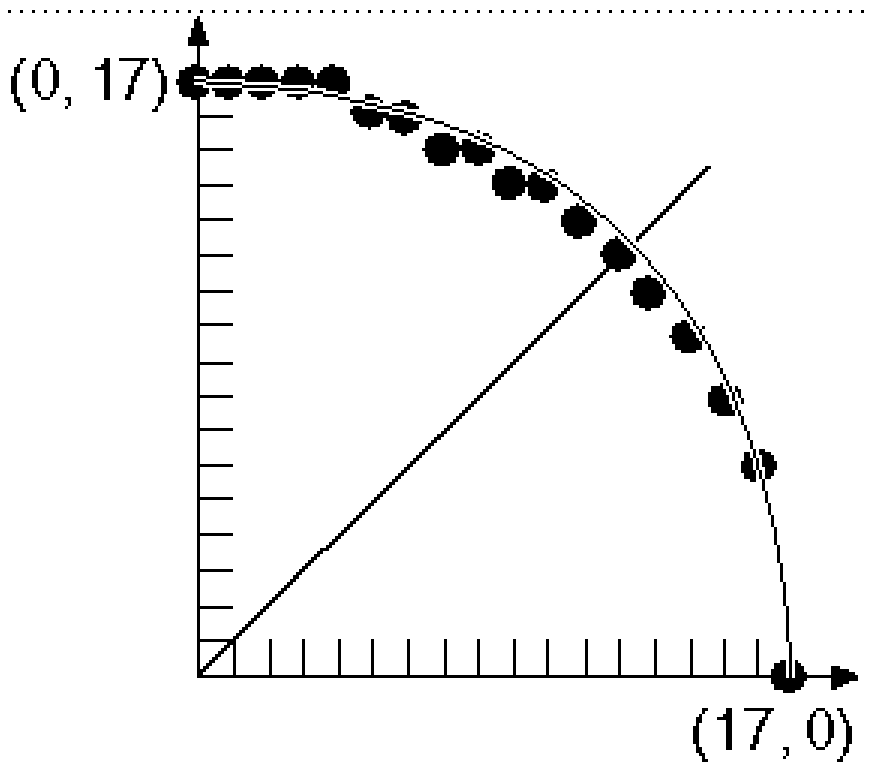


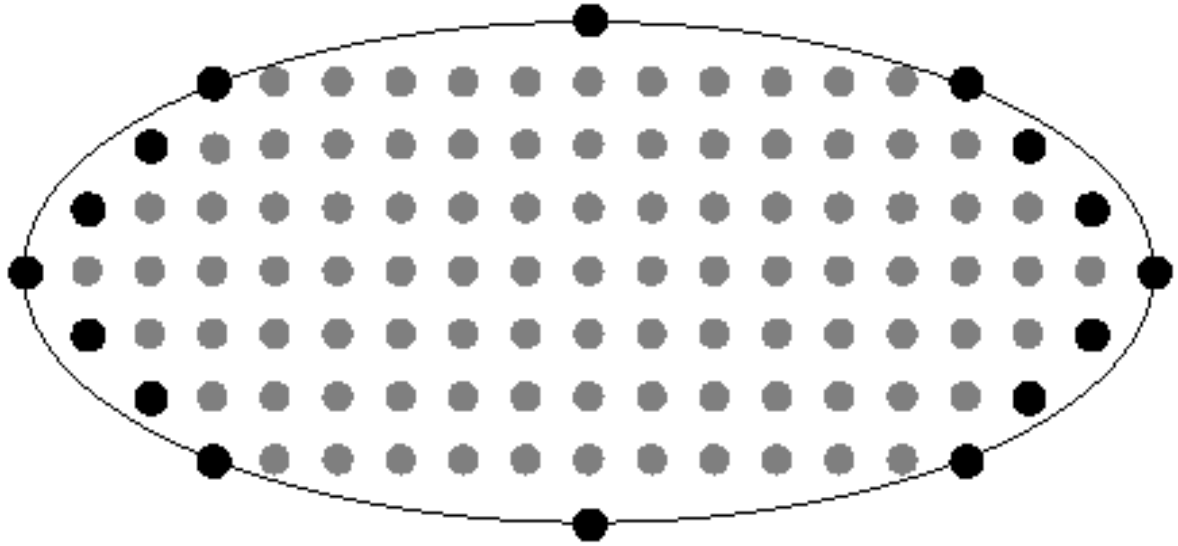
Drawing Filled Polygons

1. Find intersection of scanline with polygon edges
2. Sort intersections by increasing x
3. Fill the polygon between pairs of intersections (spans)

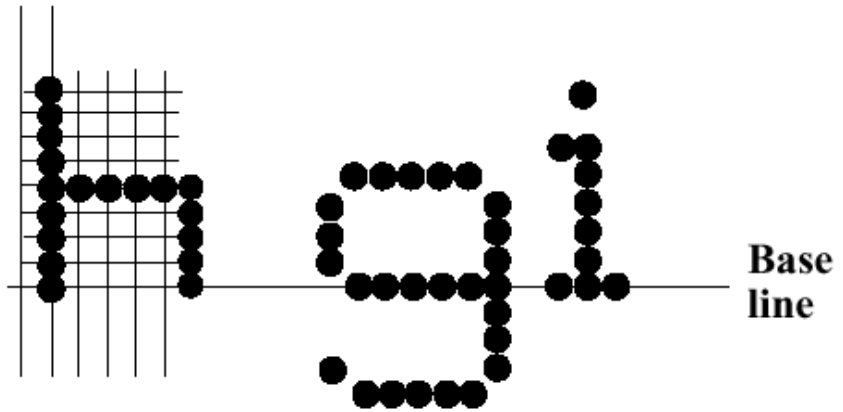


Drawing Circles and Ellipses





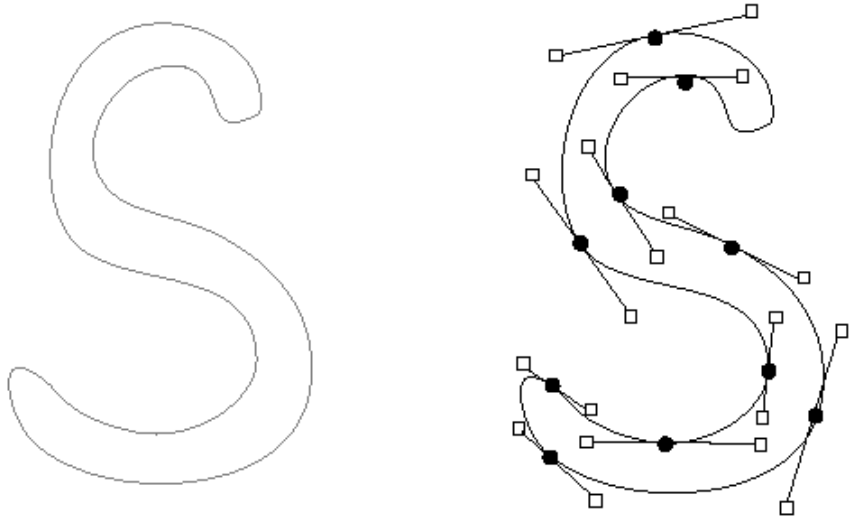
Drawing Characters



- First approach used a bitmap or "texture" in computer graphics lingo

  This has all the usual limitations in scaling that one would expect

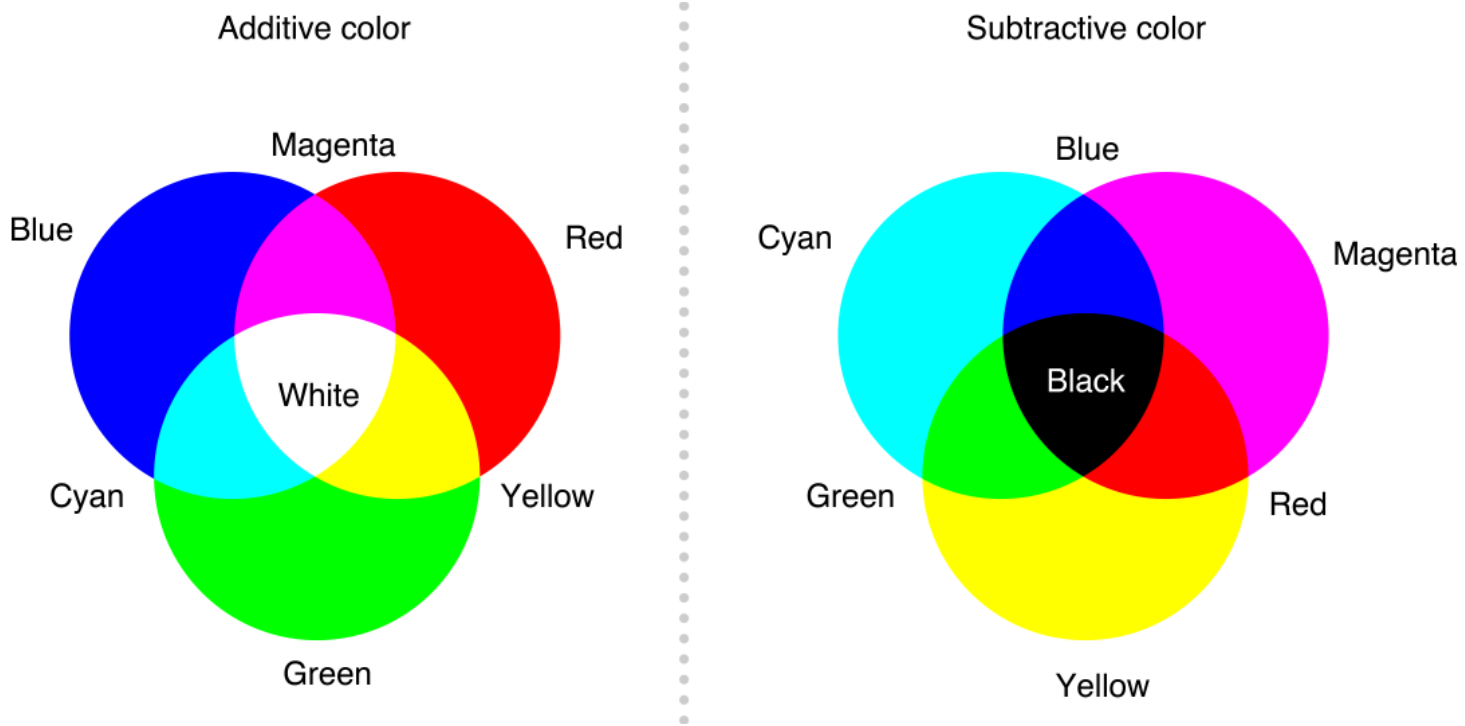
Drawing Characters



- Outline fonts: defined in terms of mathematical drawing primitives (lines, arcs, splines) and thus scalable, but more CPU intensive (e.g. Adobe PostScript™, Microsoft TrueType™)

  Font design (typography is highly skilled specialty, involving graphical and algorithmic design)

Representing Color

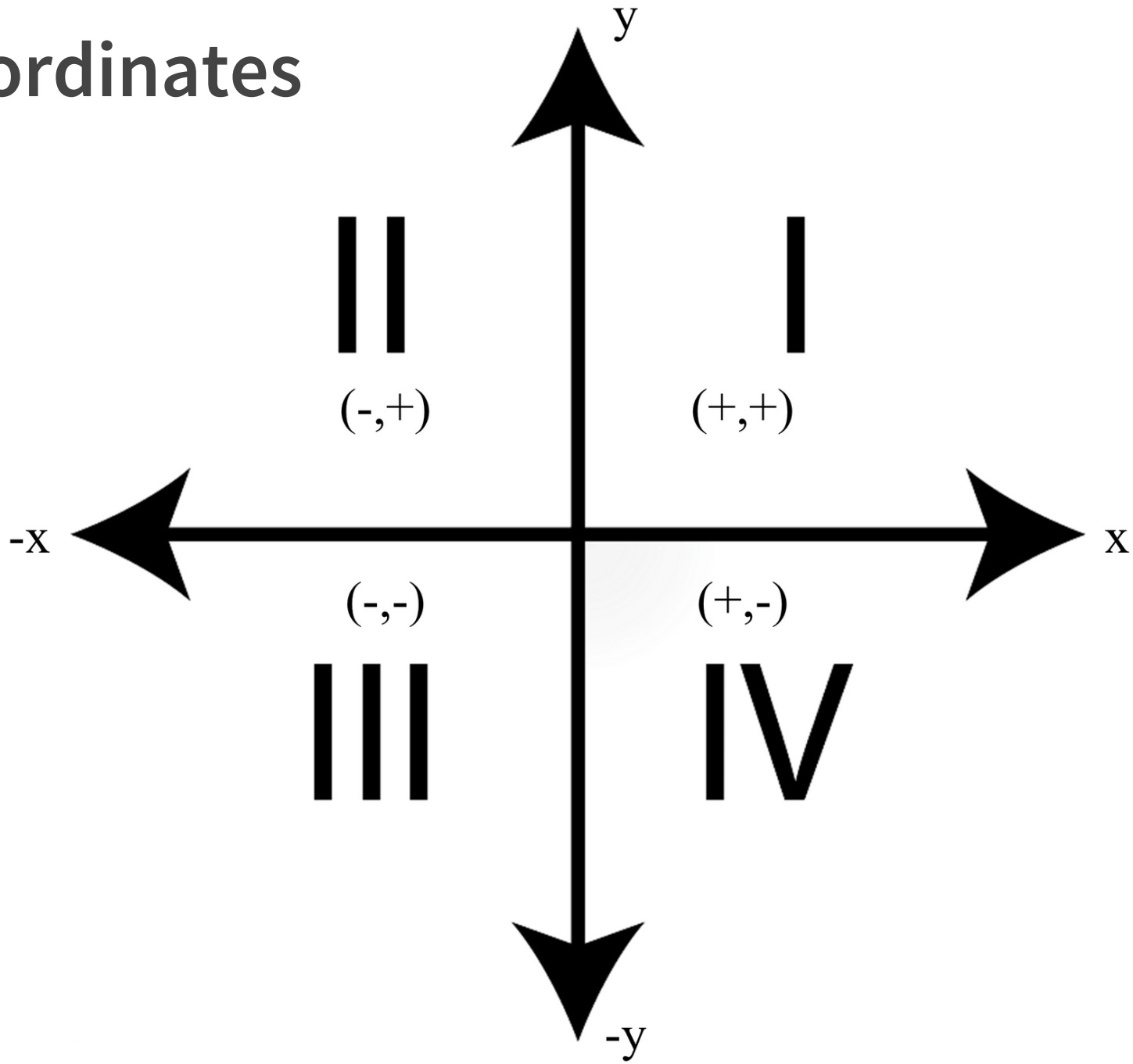


In computer graphics, we use a three-component additive color model (RGB).

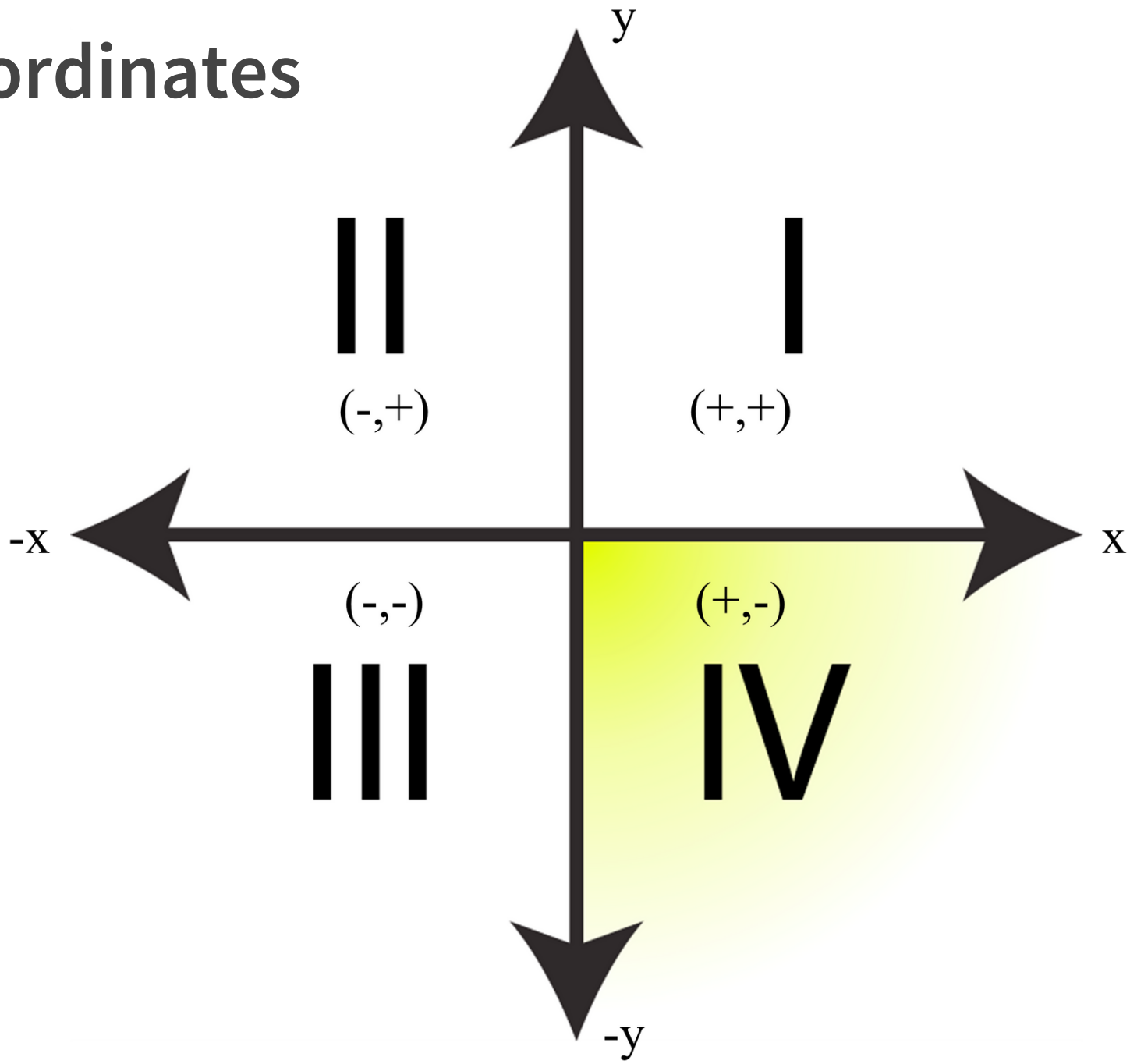
RGBA colors also include an alpha channel for transparency.

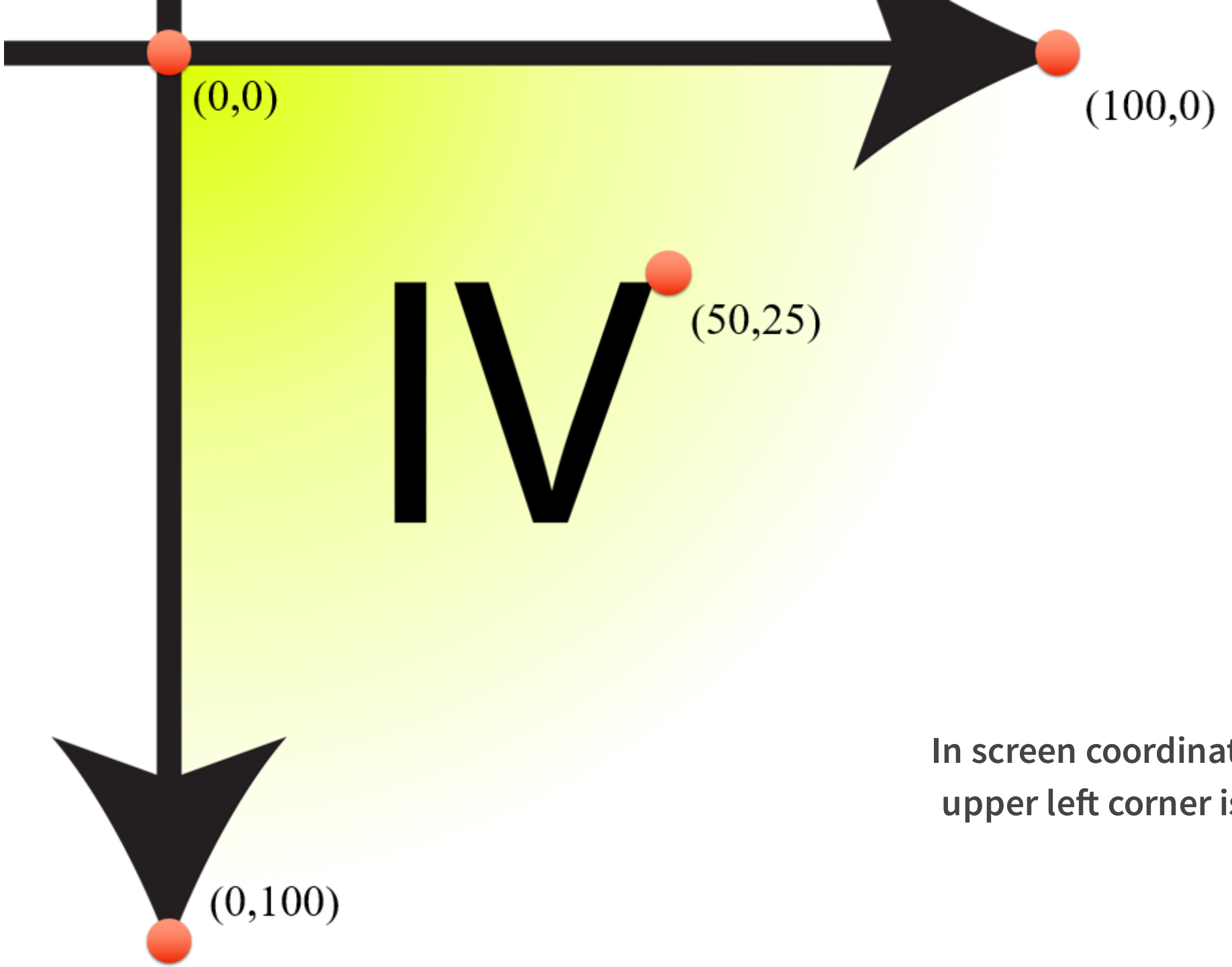
Coordinate Systems, Points, and Vectors

Screen Coordinates



Screen Coordinates

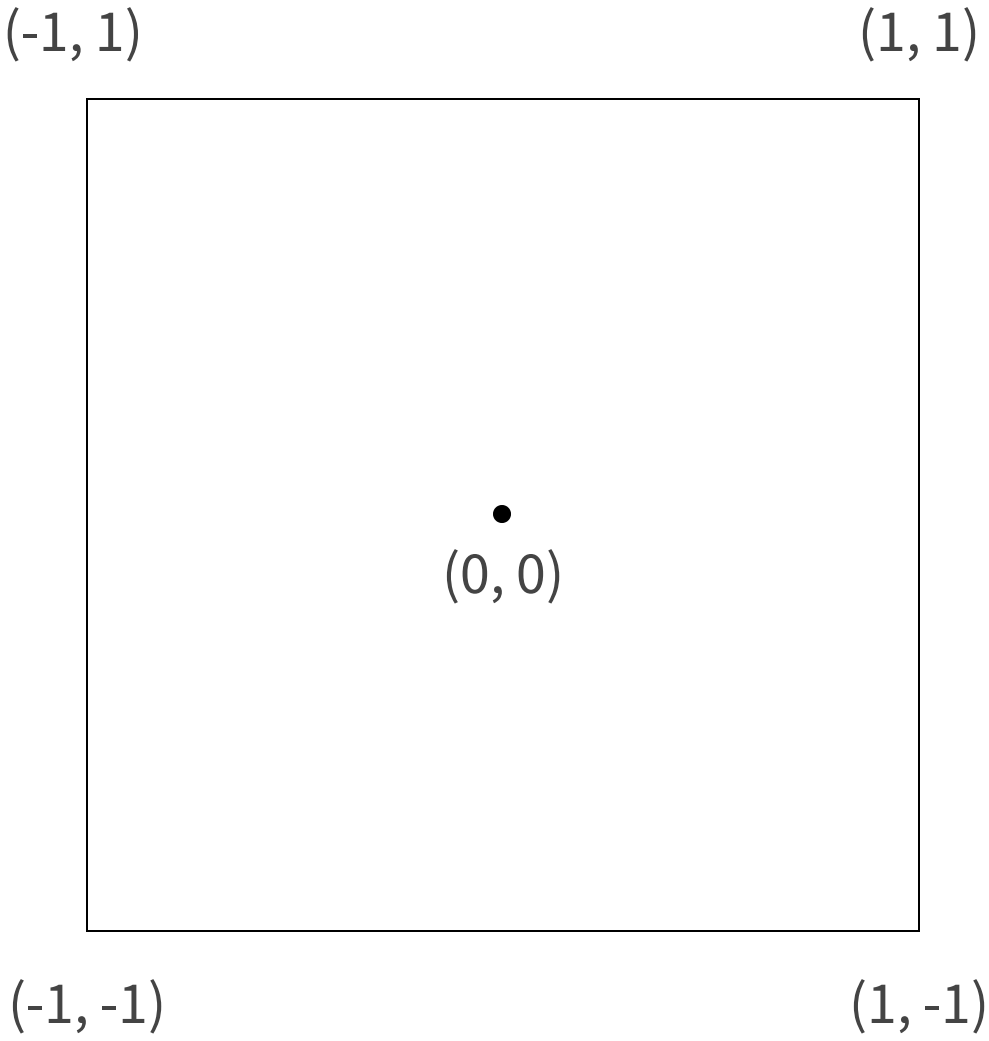




In screen coordinates, the upper left corner is (0,0)!

Normalized Device Coordinates

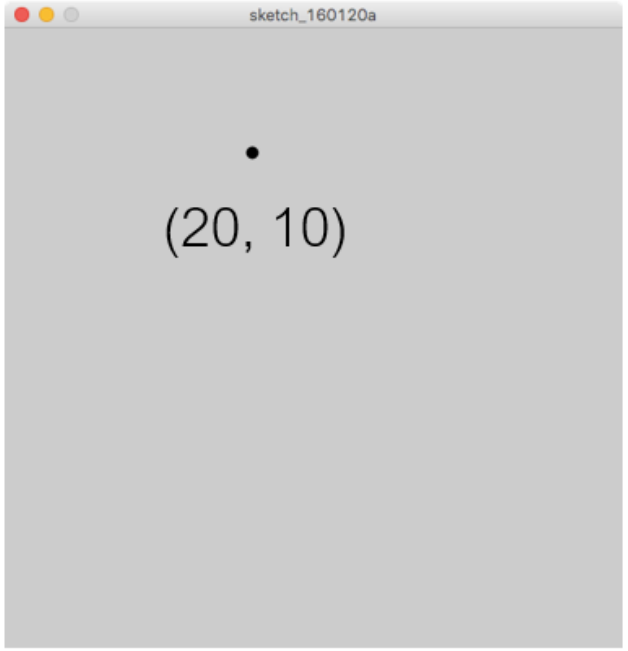
- Graphics programs may be run with different screen resolutions and window sizes.
- We therefore need a screen independent coordinate system.
- This is the 2D coordinate system you will be using in the programming assignment.
- Note that mouse events are reported in screen coordinates, so they will need to be converted!

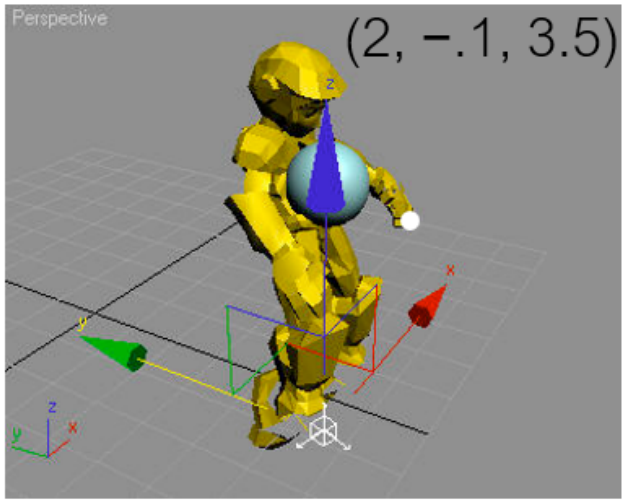


Points

A point is a location in space (2D, 3D, etc.)

It can be specified by a tuple of numbers, relative to a coordinate system.





Points

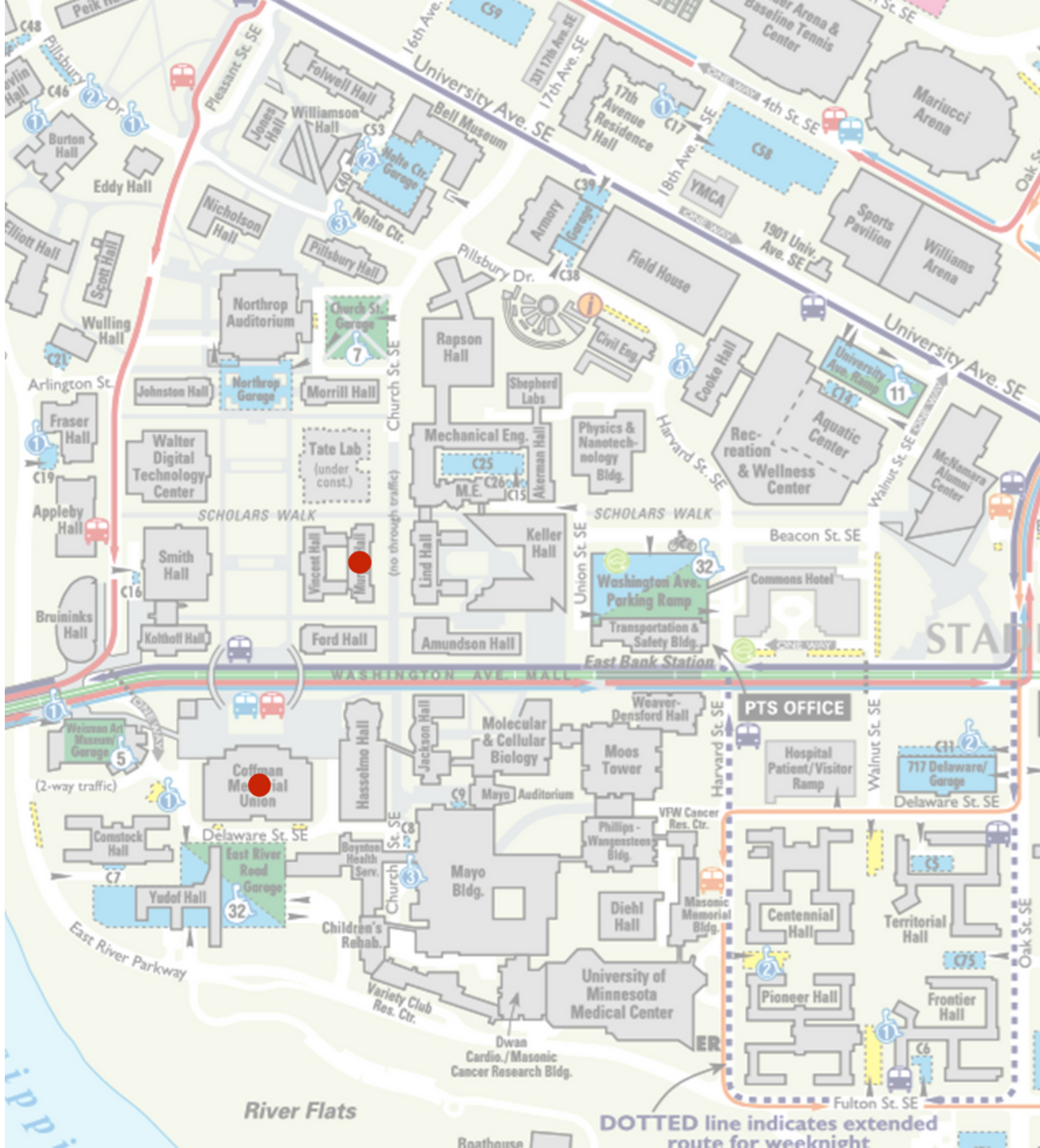
Points identify a position in space, but what else can they do?

It doesn't really make sense to add or multiply points.

Murphy + Coffman = ?

1.4 x Coffman = ?

... but maybe you can subtract them?



Subtracting Points to Get Vectors

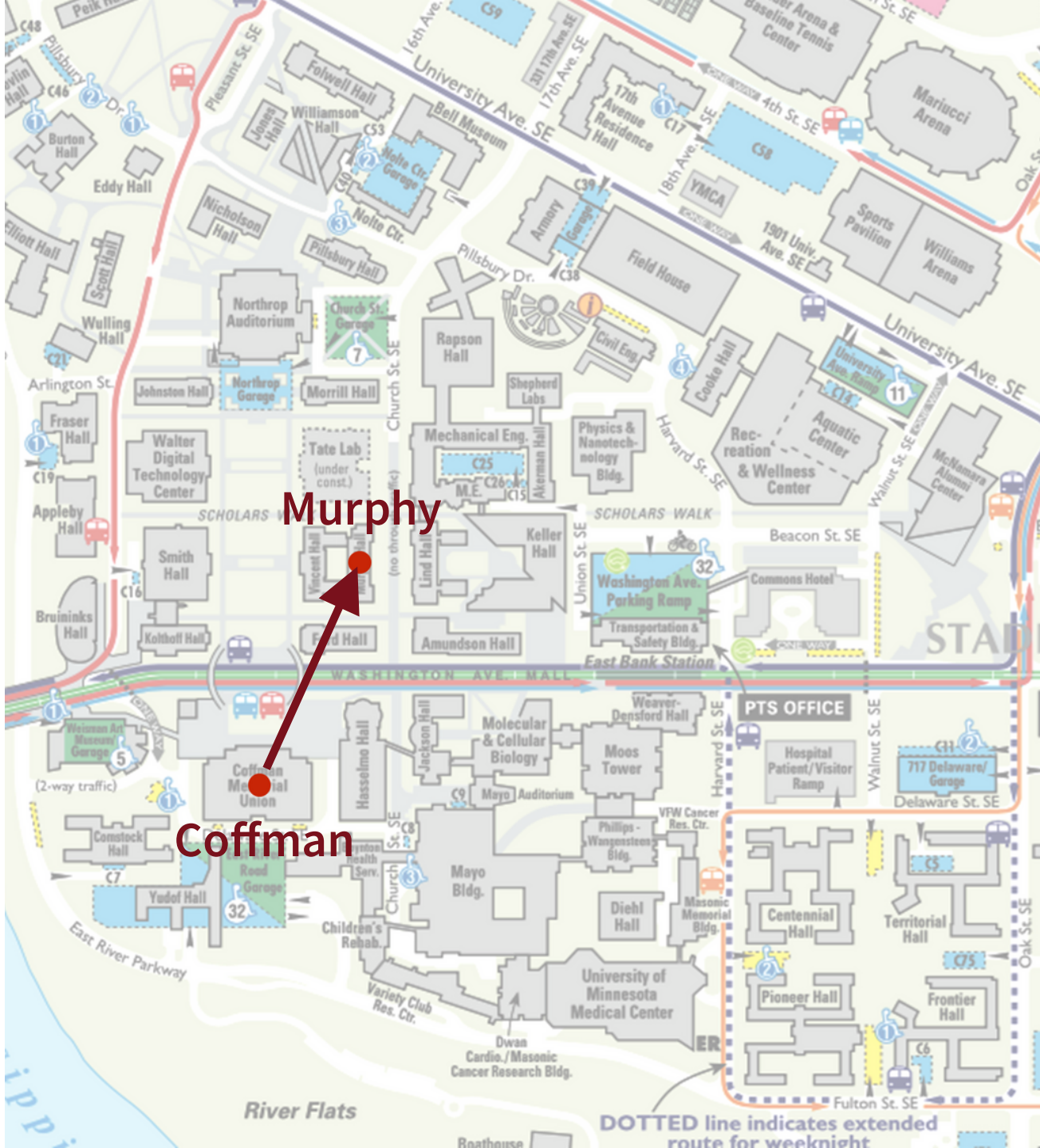
$$\underset{\text{point}}{(2,0)} - \underset{\text{point}}{(5,0)} = \underset{\text{vector}}{(-3,0)}$$

The difference between (2,0) and (5,0) is the direction and distance to travel to get to (2,0) from the starting point of (5,0).

Real-ish Example

Murphy - Coffman =

direction and distance to travel from Coffman to Murphy

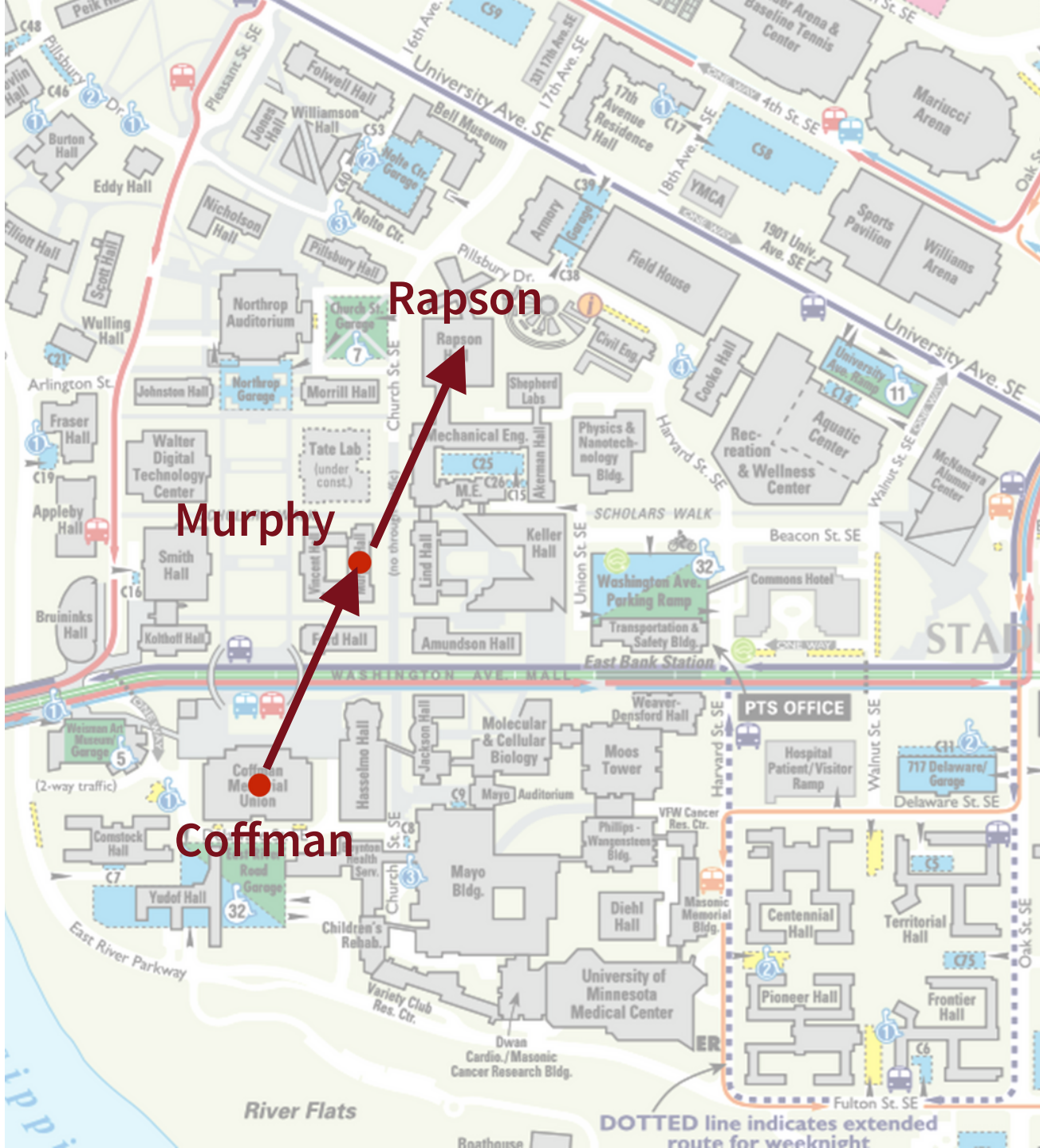


Real-ish Example

Let ***v*** = (Murphy - Coffman)

Coffman + ***v*** = Murphy

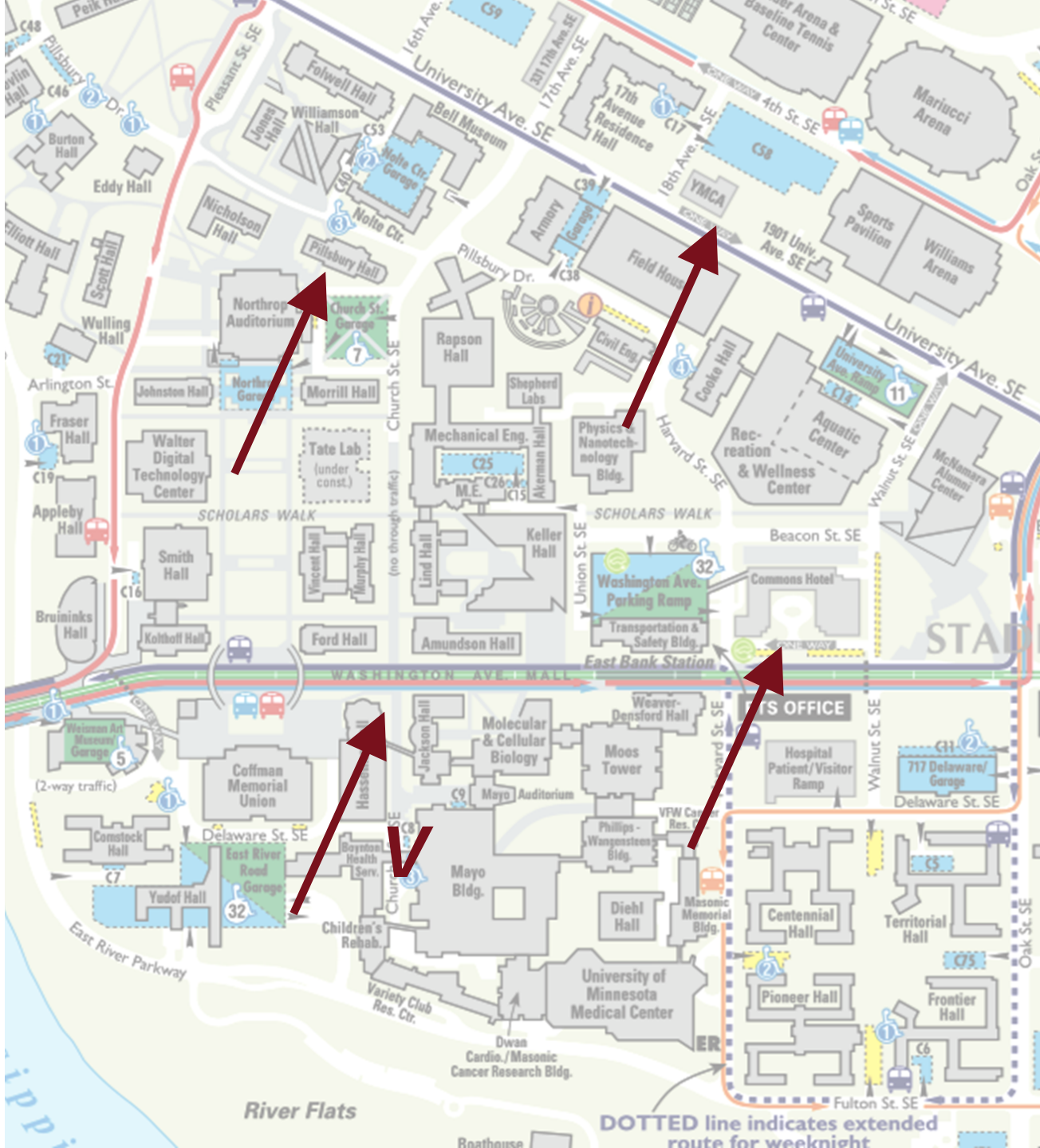
Murphy + ***v*** = Rapson



Vectors

Vectors have length and direction, but **no fixed position.**

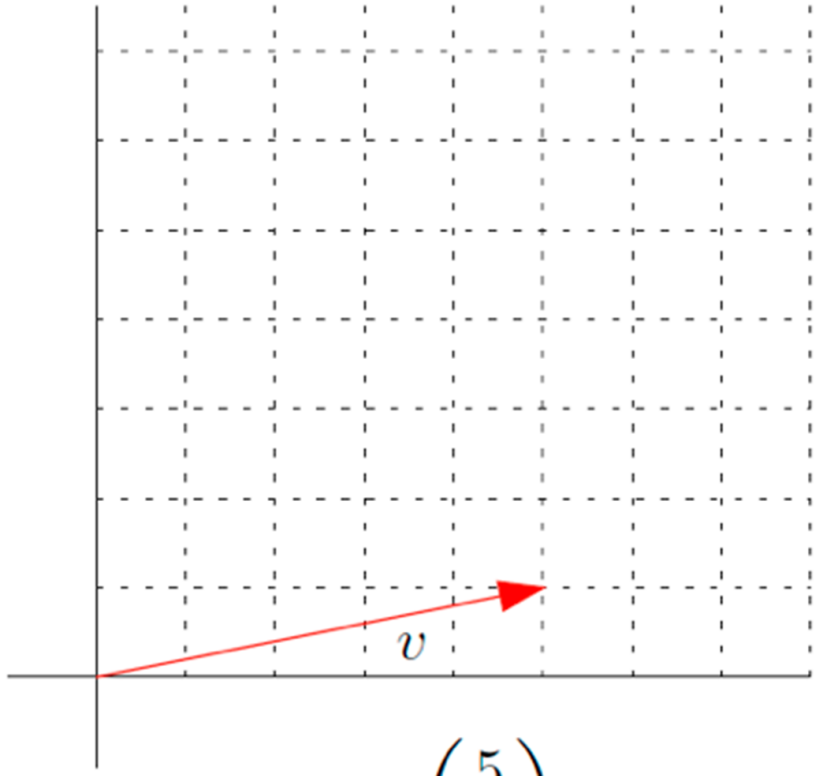
Can be added, subtracted, and scaled.



Vectors

Given a coordinate system, we can express a vector as a list of numbers.

$$v = \begin{pmatrix} v_1 \\ v_2 \\ \vdots \\ v_d \end{pmatrix}$$



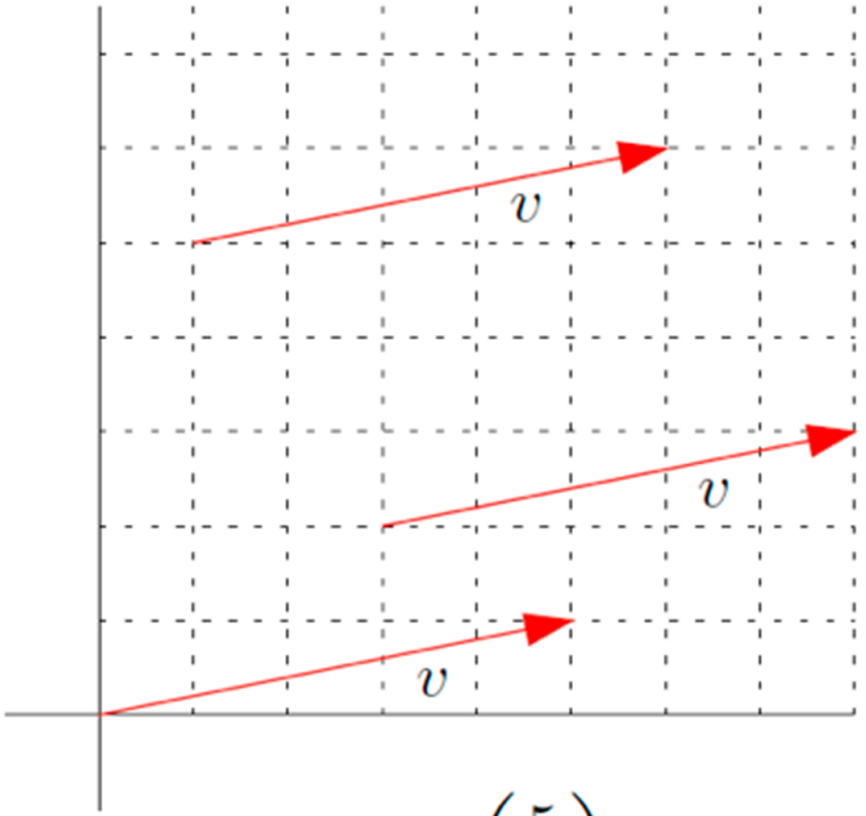
$$v = \begin{pmatrix} 5 \\ 1 \end{pmatrix}$$

Vectors

A vector can be seen as:

- an offset from the origin
- a little arrow

These are all the same vector.

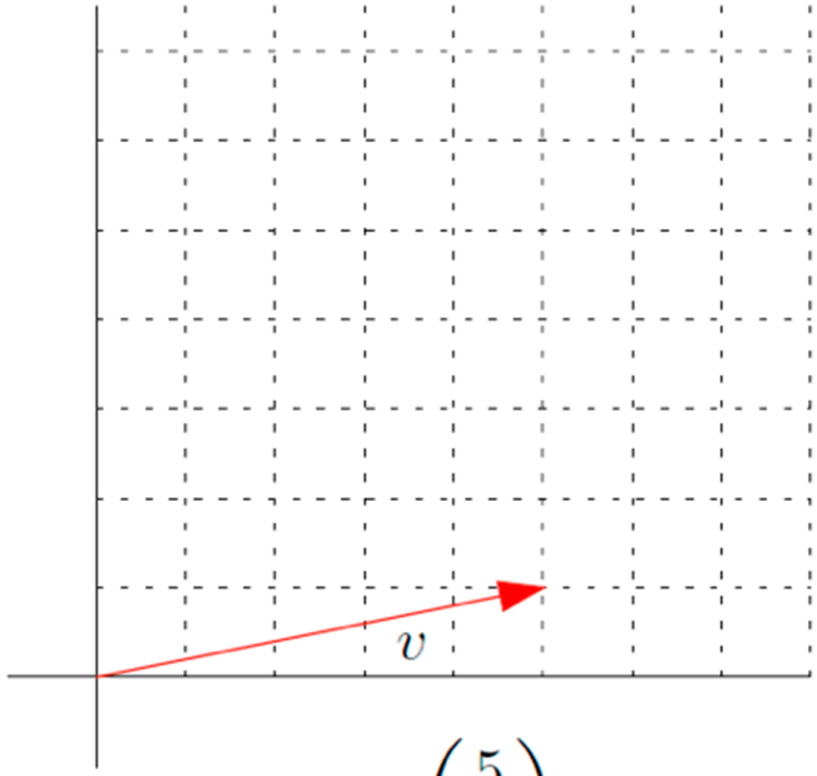


$$v = \begin{pmatrix} 5 \\ 1 \end{pmatrix}$$

Length (Magnitude)

A vector has a **length**, denoted $||\boldsymbol{v}||$

$$\|v\| = \sqrt{v_1^2 + v_2^2 + \cdots + v_d^2}$$



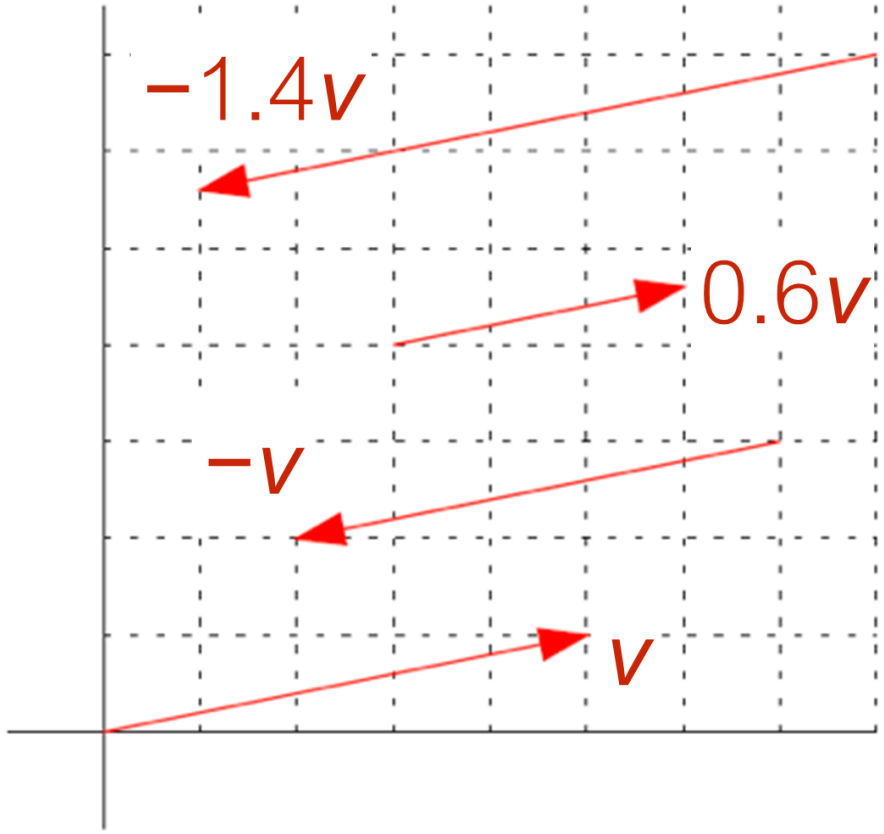
$$v = \begin{pmatrix} 5 \\ 1 \end{pmatrix}$$

Scalar Multiplation

Multiplying a vector $\boldsymbol{v}$ by a scalar (real number) $\boldsymbol{c}$ gives a new vector,

$\boldsymbol{cv} = (\boldsymbol{cv}1, \boldsymbol{cv}2, \ldots, \boldsymbol{cv}d)$

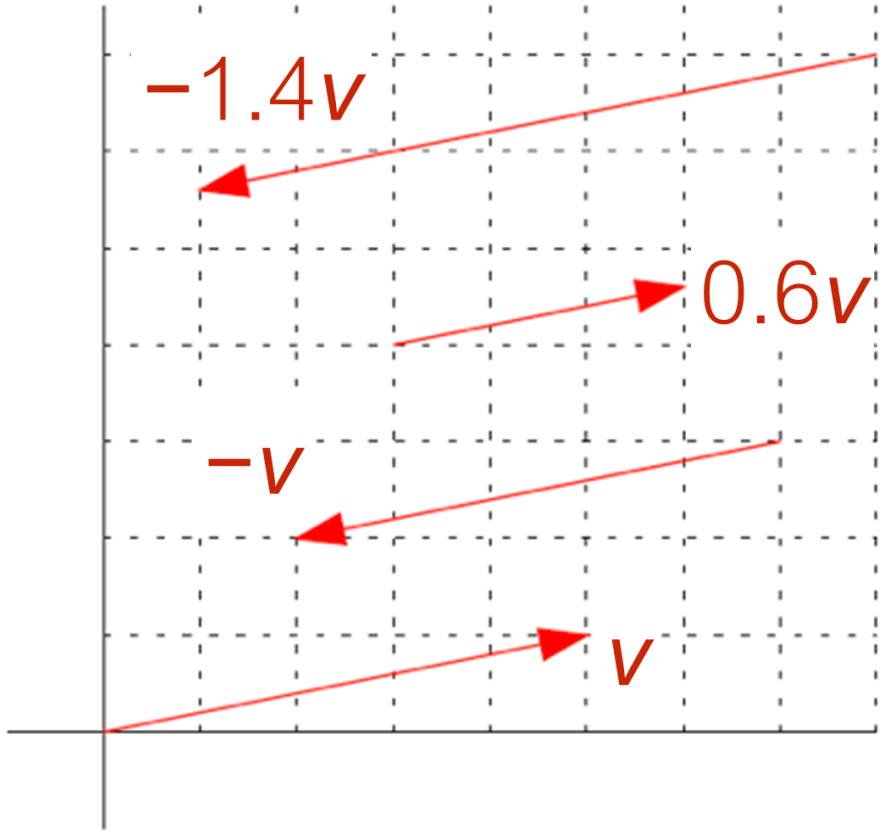
Note that $\boldsymbol{cv}$ has either the same or the opposite direction as $\boldsymbol{v}$.



Unit Vectors

A vector v is a **unit vector** if $\|\boldsymbol{v}\| = 1$.

Normalizing a vector means finding a unit vector strictly parallel to it. How might you do this?

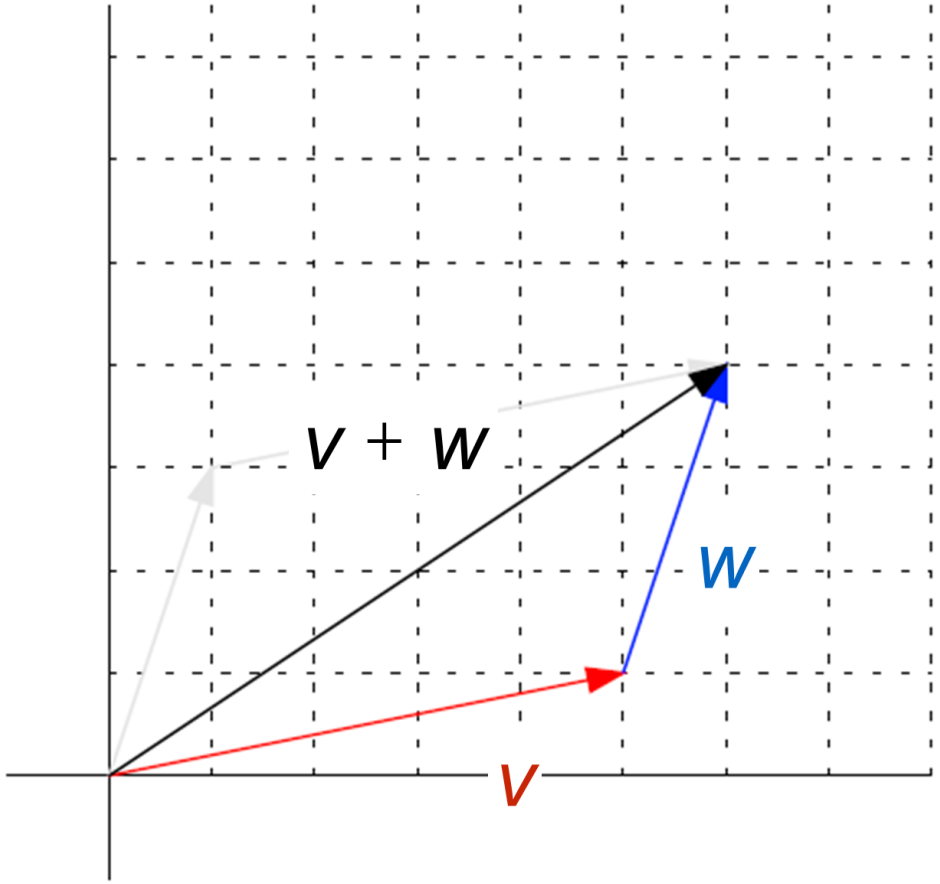


Vector Addition

To add two vectors $\boldsymbol{v}$ and $\boldsymbol{w}$:

$\boldsymbol{v} + \boldsymbol{w} = (\boldsymbol{v}1+\boldsymbol{w}1, \boldsymbol{v}2+\boldsymbol{w}2, \ldots)$

Geometrically, this puts one vector after the other.



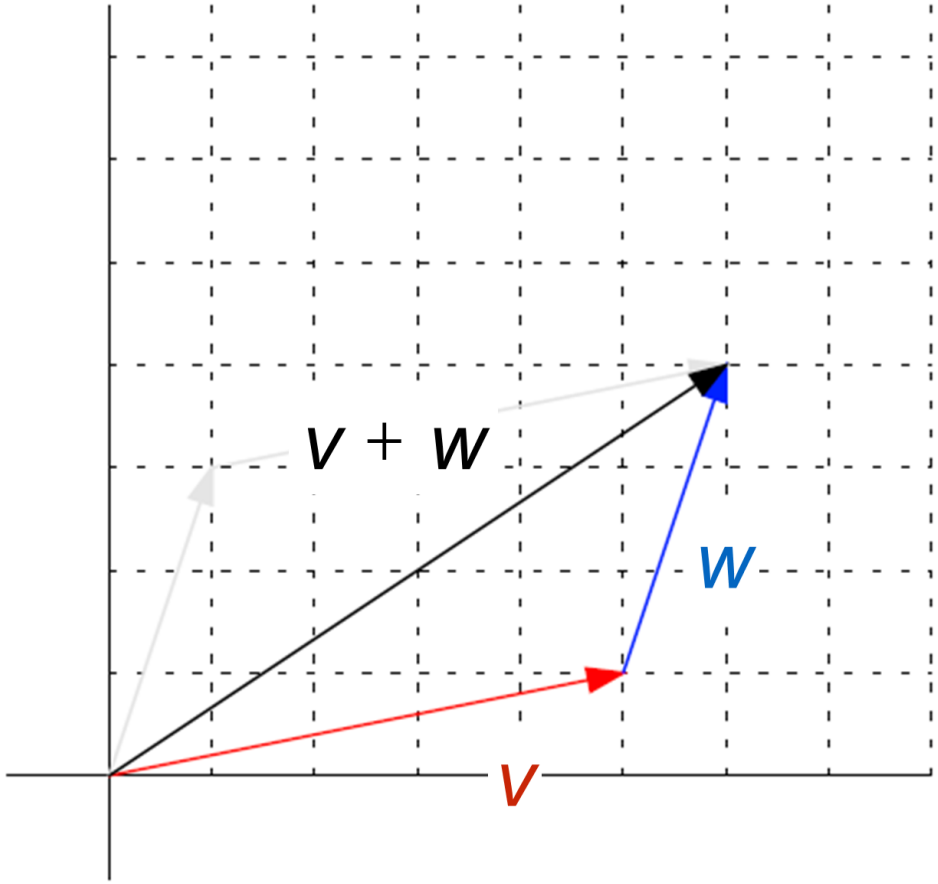
Vector Subtraction

To add two vectors ***v*** and ***w***:

$\boldsymbol{v} + \boldsymbol{w} = (\boldsymbol{v}1+\boldsymbol{w}1, \boldsymbol{v}2+\boldsymbol{w}2, \ldots)$

Geometrically, this puts one vector after the other.

How would you define vector subtraction?



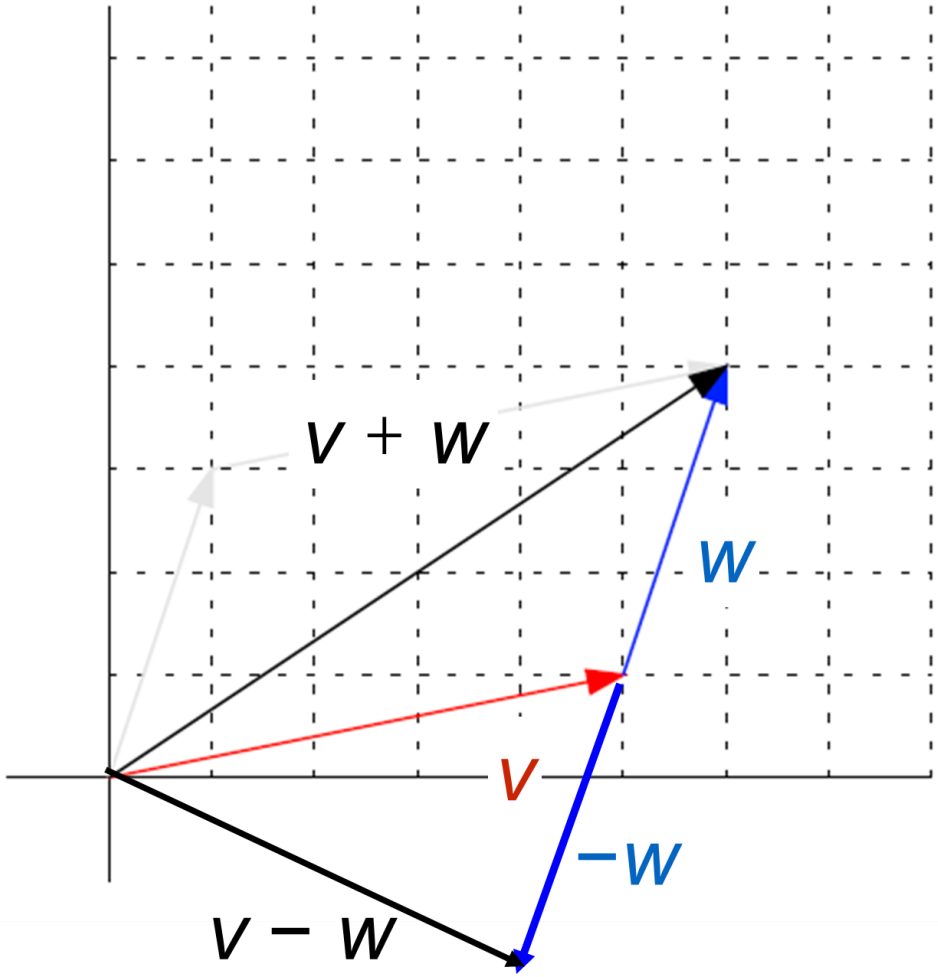
Vector Subtraction

To add two vectors $\boldsymbol{v}$ and $\boldsymbol{w}$:

$\boldsymbol{v} + \boldsymbol{w} = (\boldsymbol{v}1+\boldsymbol{w}1, \boldsymbol{v}2+\boldsymbol{w}2, \ldots)$

Geometrically, this puts one vector after the other.

How would you define vector subtraction?



Assignment 1 Overview

- Public GitHub template repo (do not clone this)
  https://github.com/CSCI-4611-Fall-2022/Assignment-2

- The GitHub classroom link to create your private repo is posted under Assignment 1 on Canvas

- On Thursday, we will have a live programming class where I will implement and explain the starter code