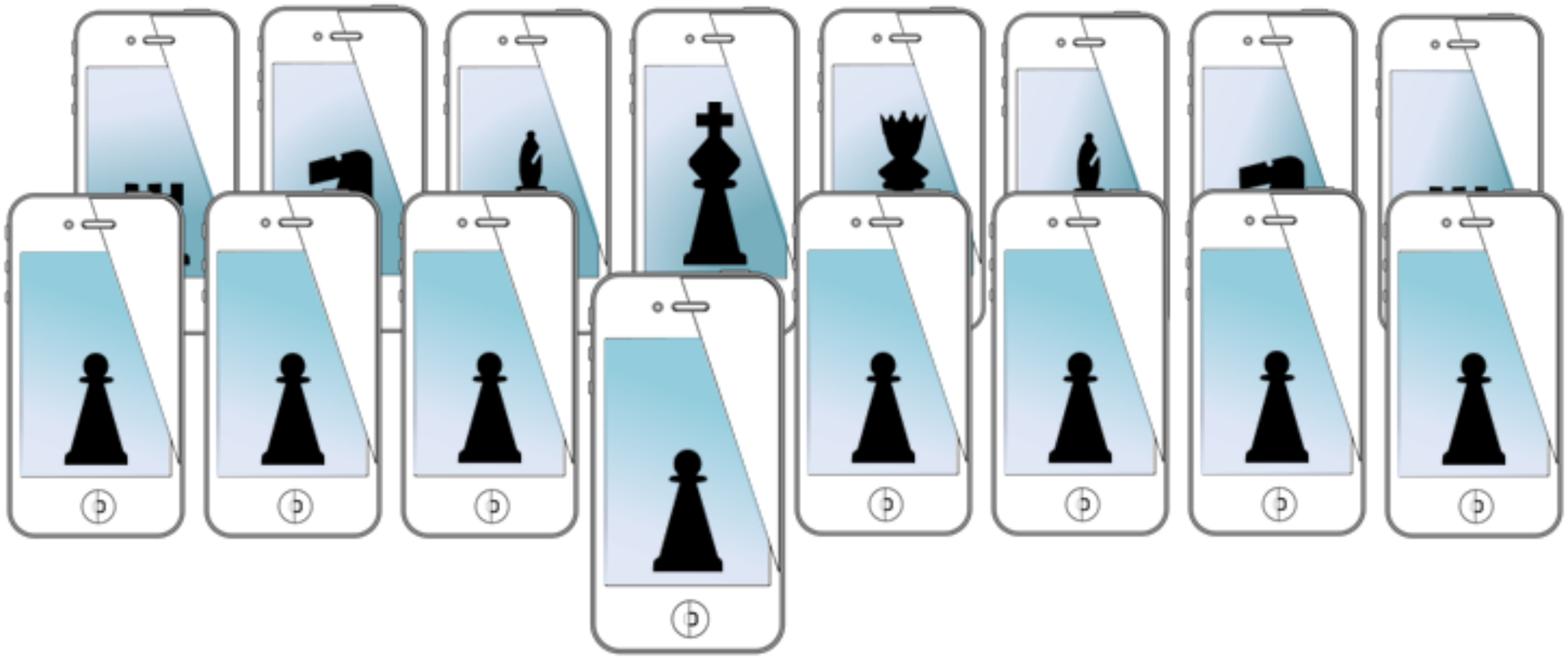
MOBILE SENSING LEARNING & CONTROL



CSE5323 & 7323

Mobile Sensing, Learning, and Control

lecture ten: core motion demo and image processing

Eric C. Larson, Lyle School of Engineering,
Computer Science and Engineering, Southern Methodist University

course logistics

- A2 grades will be up sometime this week
- A3 will be due Monday, March 3rd, 6PM
  - see updated schedule

course logistics

- A2 grades will be up sometime this week
- A3 will be due Monday, March 3rd, 6PM
  - see updated schedule

Create an iOS application using the MotionExample template that:

- Displays the number of steps a user has walked today and the previous day
- Displays a realtime count of the number of steps a users has taken today
- Displays the number of steps until the user reaches a (user settable) daily goal
- Displays the current activity of the user: {still, walking, running, in car}
- Uses {acceleration, gyro, fused motion} to distinguish when the user is climbing stairs
- Estimates the number of stairs climbed (coarse estimate)
  - this does NOT need to count stair steps in the background
- (extra credit, up to 0.5 points) add a feature that vibrates the motor and infers pressure of the grip based on the accelerometer and motor

The application should make use of the M7 co-processor whenever possible.

course logistics

- A2 grades will be up sometime this week
- A3 will be due Monday, March 3rd, 6PM
  - see updated schedule

Create an iOS application using the MotionExample template that:

- Displays the number of steps a user has walked today and the previous day
- Displays a realtime count of the number of steps a users has taken today
- Displays the number of steps until the user reaches a (user settable) daily goal
- Displays the current activity of the user: {still, walking, running, in car}
- Uses {acceleration, gyro, fused motion} to distinguish when the user is climbing stairs
- Estimates the number of stairs climbed (coarse estimate)
  - this does NOT need to count stair steps in the background
- (extra credit, up to 0.5 points) add a feature that vibrates the motor and infers pressure of the grip based on the accelerometer and motor

The application should make use of the M7 co-processor whenever possible.

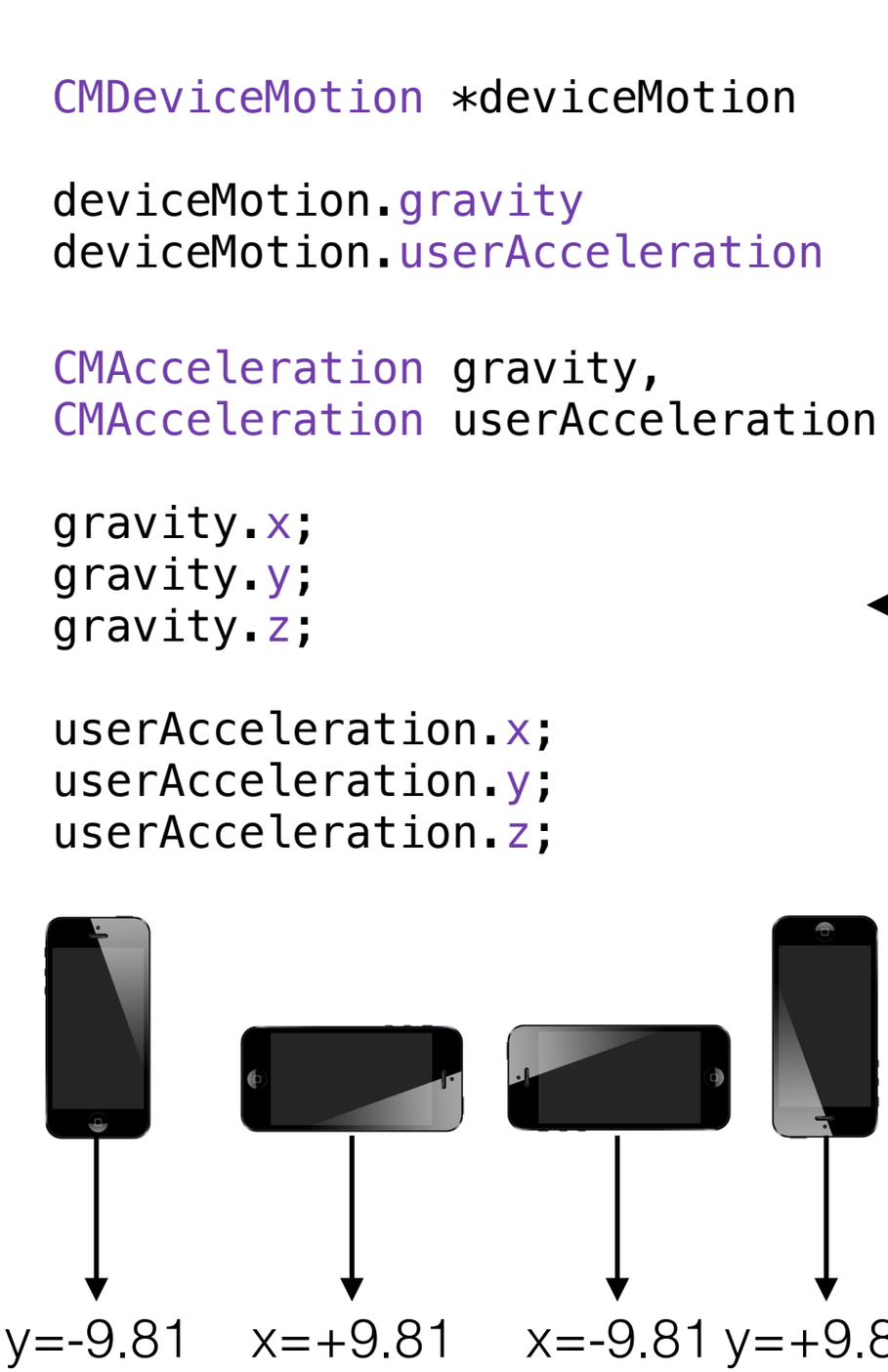
make the user interface engaging!

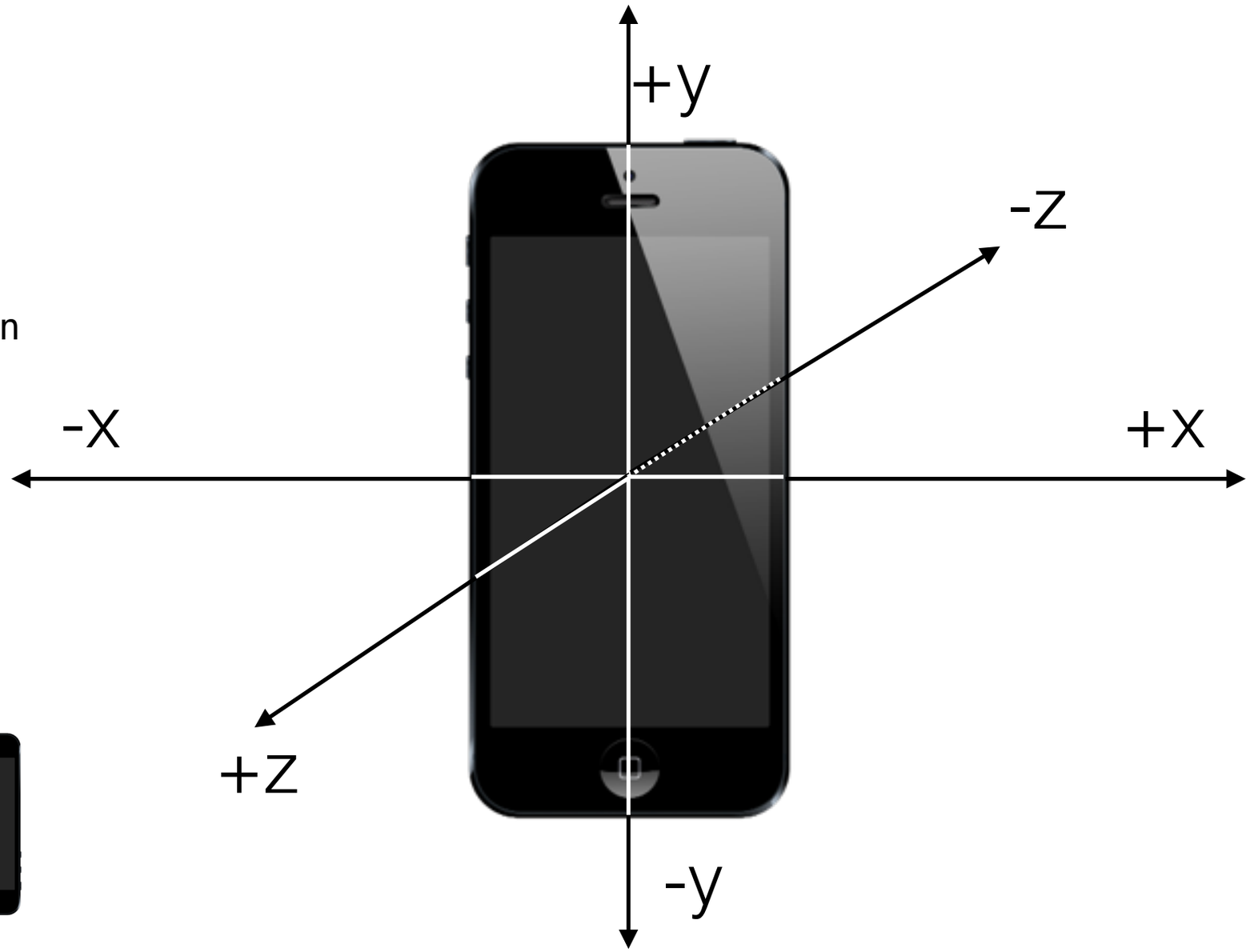
agenda

- review motion sensors
- finish core motion demos
  - graphing
- image processing basics

accessing the accelerometer

```
CMDeviceMotion *deviceMotion

deviceMotion.gravity
deviceMotion.userAcceleration

CMAcceleration gravity,
CMAcceleration userAcceleration

gravity.x;
gravity.y;
gravity.z;

userAcceleration.x;
userAcceleration.y;
userAcceleration.z;
```



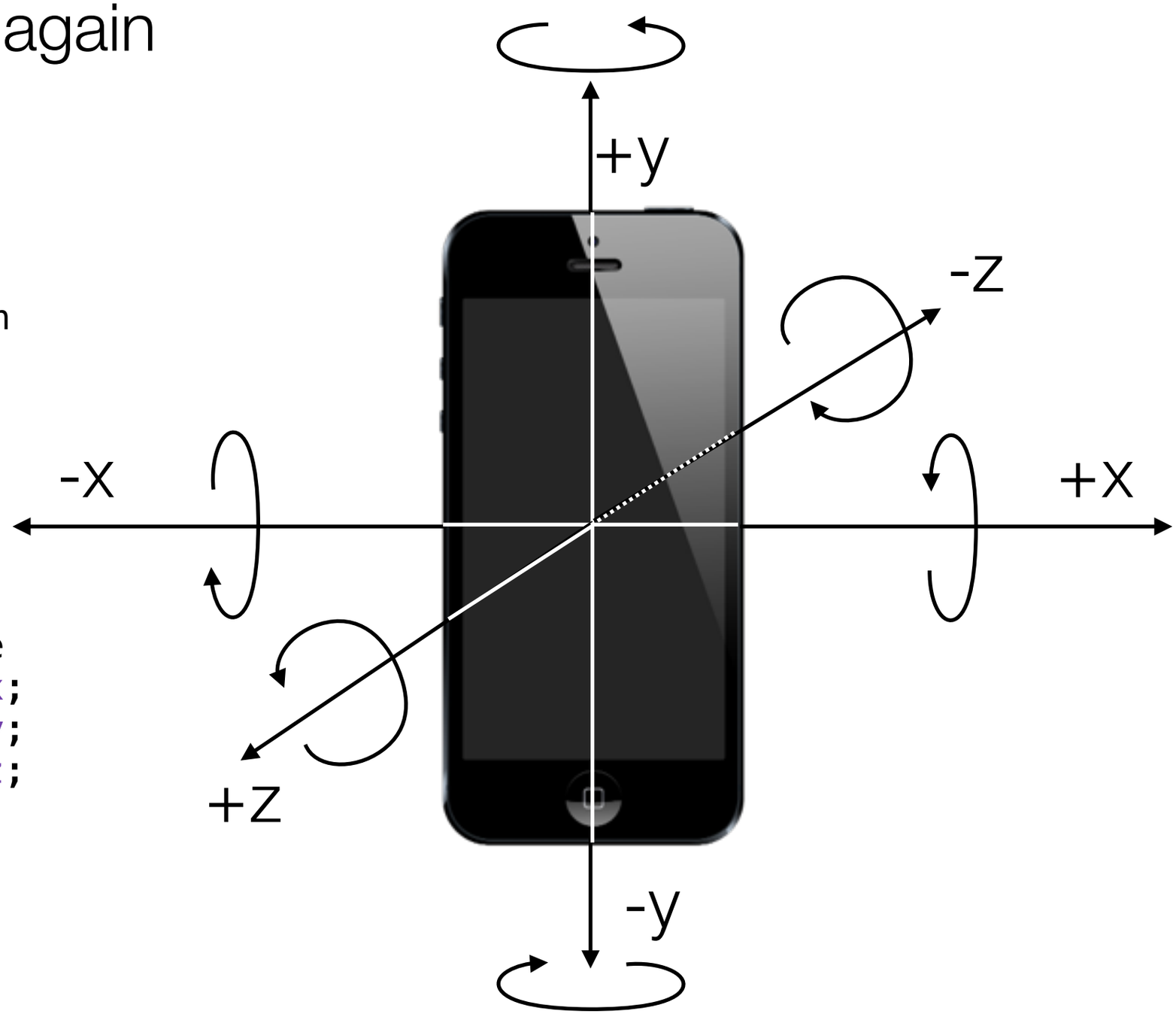


accessing the gyro

- use device motion again

```
CMDeviceMotion *deviceMotion

deviceMotion.rotationRate
```

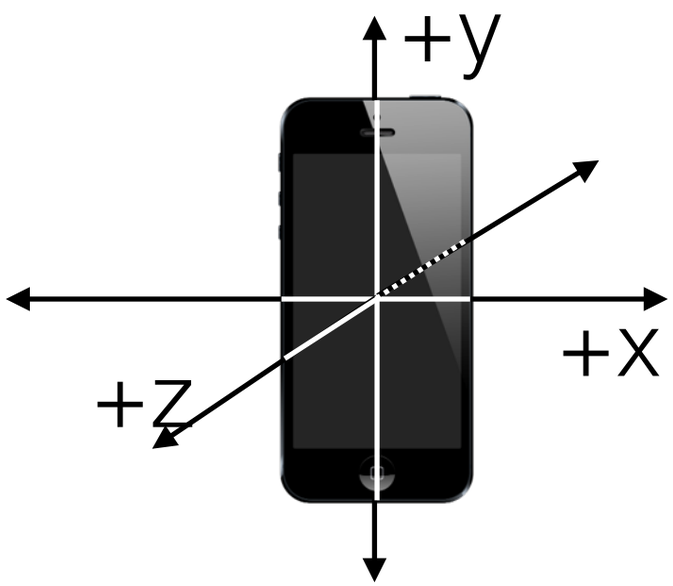
```
CMRotationRate rotationRate
rotX[head] = rotationRate.x;
rotY[head] = rotationRate.y;
rotZ[head] = rotationRate.z;
```

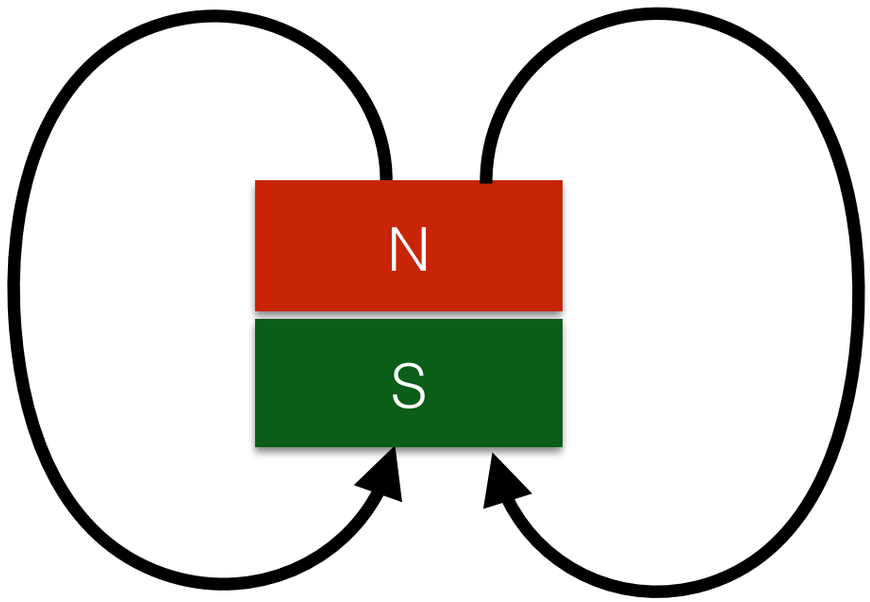


but iPhone has magnetic bias

```
CMDeviceMotion *deviceMotion

deviceMotion.magneticField
CMCalibratedMagneticField magneticField;

magneticField.field.x
magneticField.field.y
magneticField.field.z

magneticField.accuracy
```

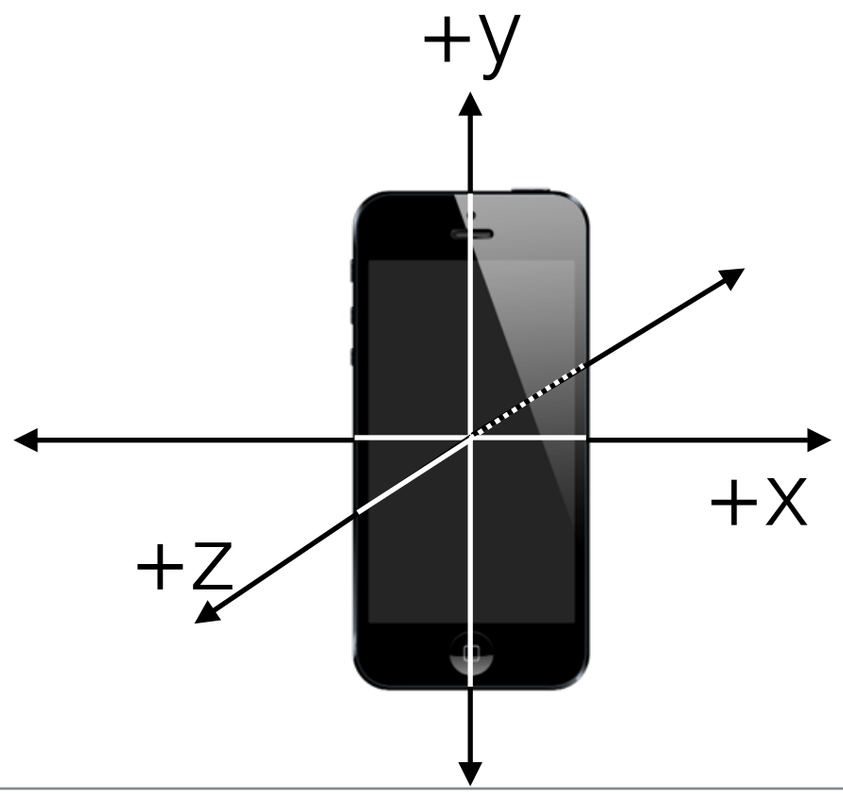
```
CMMagneticFieldCalibrationAccuracyUncalibrated = -1,
   CMMagneticFieldCalibrationAccuracyLow,
   CMMagneticFieldCalibrationAccuracyMedium,
   CMMagneticFieldCalibrationAccuracyHigh
```





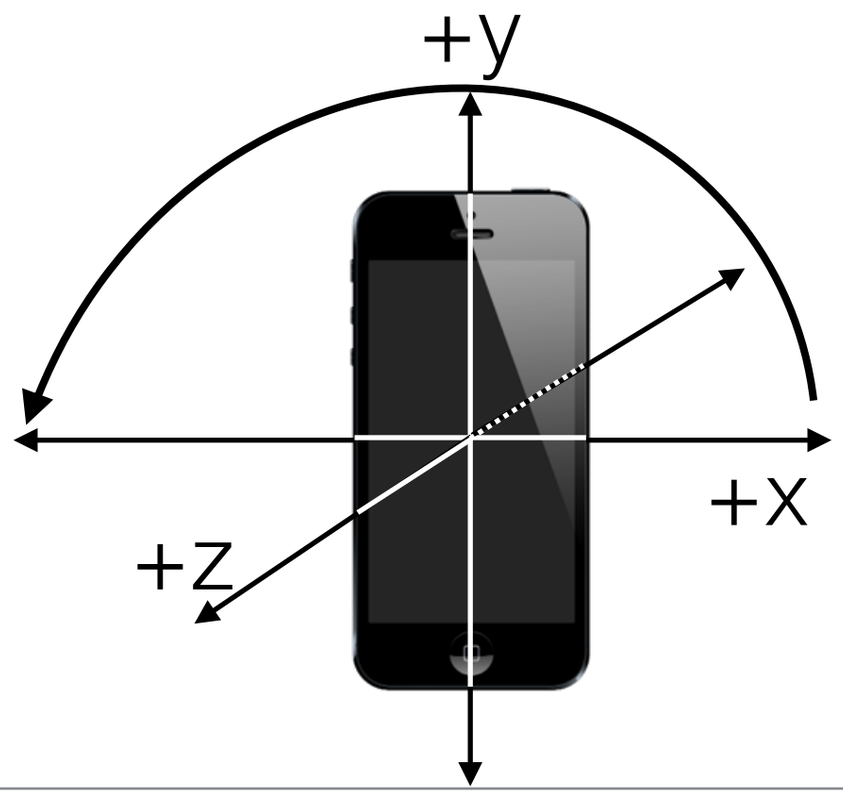
attitude

```
CMDeviceMotion *deviceMotion

deviceMotion.attitude

CMAttitude* attitude

attitude.roll;
attitude.pitch;
attitude.yaw;
```



attitude

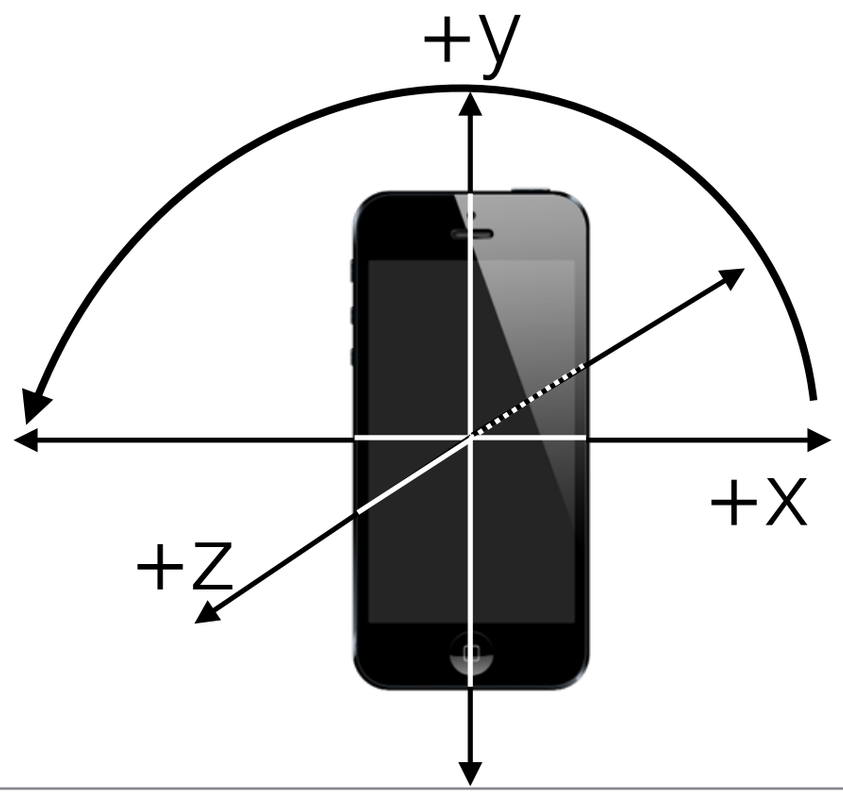
```
CMDeviceMotion *deviceMotion

deviceMotion.attitude

CMAttitude* attitude

attitude.roll;
attitude.pitch;
attitude.yaw;
```



yaw in x/y plane

attitude

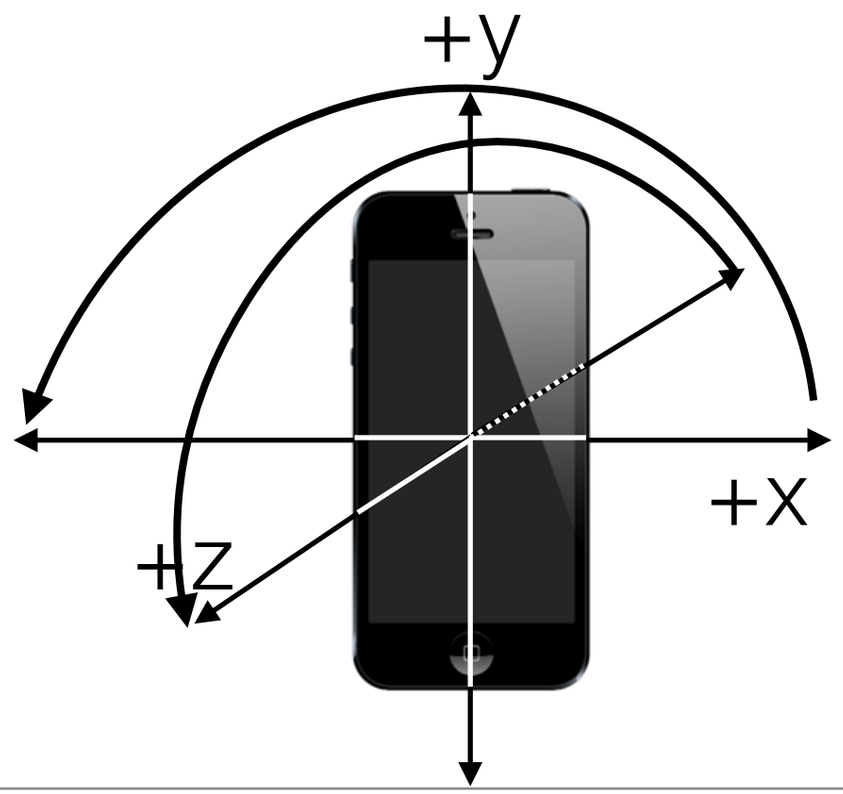
```
CMDeviceMotion *deviceMotion

deviceMotion.attitude

CMAttitude* attitude

attitude.roll;
attitude.pitch;
attitude.yaw;
```



yaw in x/y plane

attitude

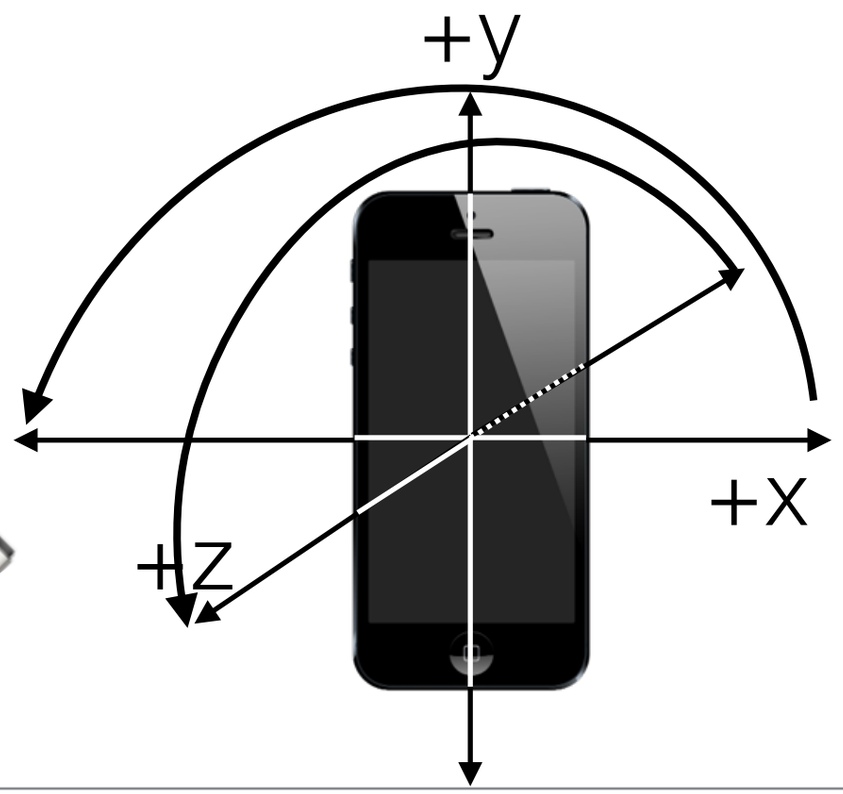
```
CMDeviceMotion *deviceMotion

deviceMotion.attitude

CMAttitude* attitude

attitude.roll;
attitude.pitch;
attitude.yaw;
```



yaw in x/y plane
pitch in y/z plane

attitude

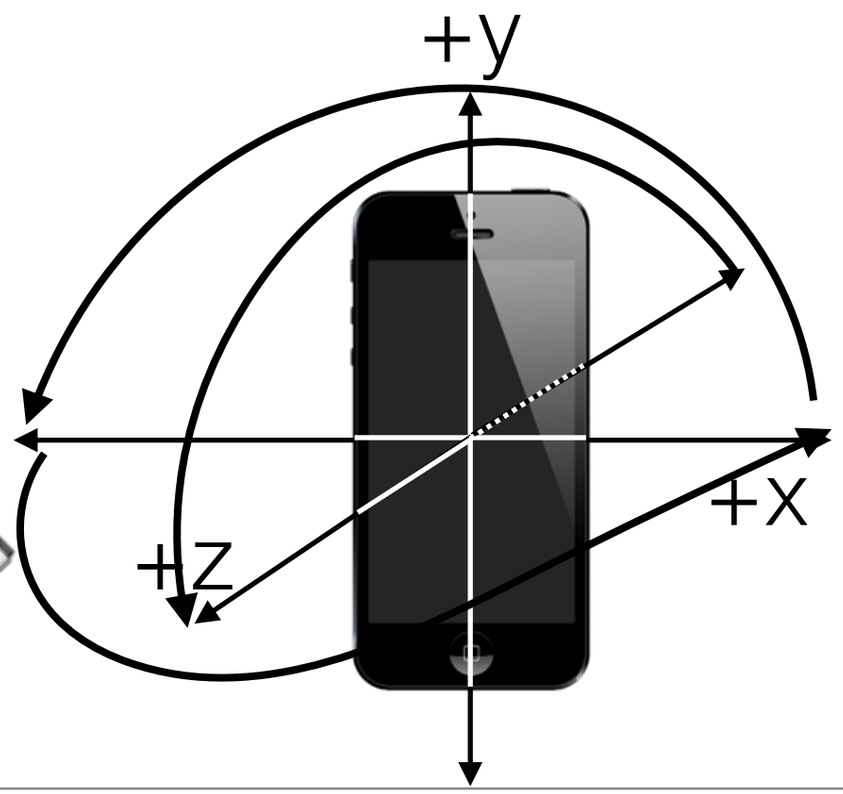
```
CMDeviceMotion *deviceMotion

deviceMotion.attitude

CMAttitude* attitude

attitude.roll;
attitude.pitch;
attitude.yaw;
```



yaw in x/y plane
pitch in y/z plane

attitude

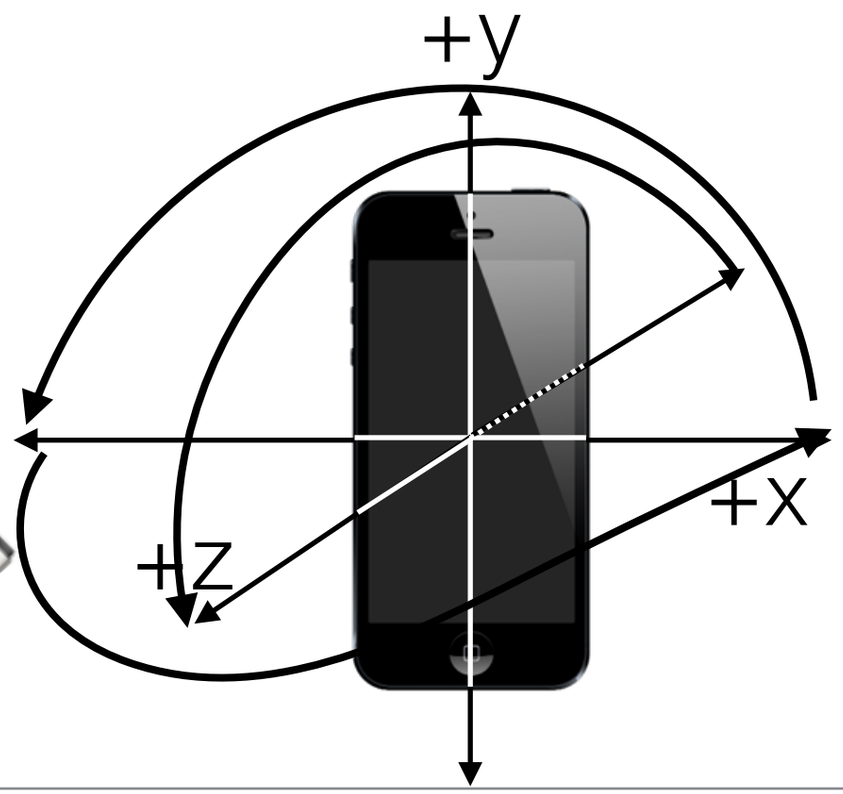
```
CMDeviceMotion *deviceMotion

deviceMotion.attitude

CMAttitude* attitude

attitude.roll;
attitude.pitch;
attitude.yaw;
```



yaw in x/y plane
pitch in y/z plane
roll in x/z plane

attitude

```
CMDeviceMotion *deviceMotion

deviceMotion.attitude

CMAttitude* attitude

attitude.roll;
attitude.pitch;
attitude.yaw;
```



yaw in x/y plane
pitch in y/z plane
roll in x/z plane

getting updates

```
// for getting access to the fused motion data (best practice, filtered)
@property (nonatomic,strong) CMMotionManager *mManager;
```

```
self.mManager = [[CMMotionManager alloc] init];

 if([self.mManager isDeviceMotionAvailable])
 {

     [self.mManager setDeviceMotionUpdateInterval:yourSamplingIntervalInSeconds];
     [self.mManager startDeviceMotionUpdatesToQueue:[NSOperationQueue mainQueue]
withHandler:^(CMDeviceMotion *deviceMotion, NSError *error) {

         //Access to all the data…
          deviceMotion.attitude,
          deviceMotion.rotationRate,
          deviceMotion.gravity,
          deviceMotion.userAcceleration,
          deviceMotion.magneticField,


     }];
 }
```

getting updates

```
// for getting access to the fused motion data (best practice, filtered)
@property (nonatomic,strong) CMMotionManager *mManager;
```

declare

```
self.mManager = [[CMMotionManager alloc] init];

 if([self.mManager isDeviceMotionAvailable])
 {

     [self.mManager setDeviceMotionUpdateInterval:yourSamplingIntervalInSeconds];
     [self.mManager startDeviceMotionUpdatesToQueue:[NSOperationQueue mainQueue]
withHandler:^(CMDeviceMotion *deviceMotion, NSError *error) {

         //Access to all the data…
          deviceMotion.attitude,
          deviceMotion.rotationRate,
          deviceMotion.gravity,
          deviceMotion.userAcceleration,
          deviceMotion.magneticField,


     }];
 }
```

getting updates

```
// for getting access to the fused motion data (best practice, filtered)
@property (nonatomic,strong) CMMotionManager *mManager;
```

instantiate

declare

```
self.mManager = [[CMMotionManager alloc] init];

 if([self.mManager isDeviceMotionAvailable])
 {

     [self.mManager setDeviceMotionUpdateInterval:yourSamplingIntervalInSeconds];
     [self.mManager startDeviceMotionUpdatesToQueue:[NSOperationQueue mainQueue]
withHandler:^(CMDeviceMotion *deviceMotion, NSError *error) {

         //Access to all the data…
          deviceMotion.attitude,
          deviceMotion.rotationRate,
          deviceMotion.gravity,
          deviceMotion.userAcceleration,
          deviceMotion.magneticField,


     }];
 }
```

getting updates

```
// for getting access to the fused motion data (best practice, filtered)
@property (nonatomic,strong) CMMotionManager *mManager;



self.mManager = [[CMMotionManager alloc] init];

 if([self.mManager isDeviceMotionAvailable])
 {

     [self.mManager setDeviceMotionUpdateInterval:yourSamplingIntervalInSeconds];
     [self.mManager startDeviceMotionUpdatesToQueue:[NSOperationQueue mainQueue]
withHandler:^(CMDeviceMotion *deviceMotion, NSError *error) {

         //Access to all the data…
          deviceMotion.attitude,
          deviceMotion.rotationRate,
          deviceMotion.gravity,
          deviceMotion.userAcceleration,
          deviceMotion.magneticField,


     }];
 }
```

declare

instantiate

if device is capable

getting updates

```
// for getting access to the fused motion data (best practice, filtered)
@property (nonatomic,strong) CMMotionManager *mManager;



self.mManager = [[CMMotionManager alloc] init];

 if([self.mManager isDeviceMotionAvailable])
 {

     [self.mManager setDeviceMotionUpdateInterval:yourSamplingIntervalInSeconds];
     [self.mManager startDeviceMotionUpdatesToQueue:[NSOperationQueue mainQueue]
withHandler:^(CMDeviceMotion *deviceMotion, NSError *error) {

         //Access to all the data…
          deviceMotion.attitude,
          deviceMotion.rotationRate,
          deviceMotion.gravity,
          deviceMotion.userAcceleration,
          deviceMotion.magneticField,


     }];
 }
```

declare

instantiate

if device is capable

how often to push updates

getting updates

```
// for getting access to the fused motion data (best practice, filtered)
@property (nonatomic,strong) CMMotionManager *mManager;



self.mManager = [[CMMotionManager alloc] init];

 if([self.mManager isDeviceMotionAvailable])
 {

     [self.mManager setDeviceMotionUpdateInterval:yourSamplingIntervalInSeconds];
     [self.mManager startDeviceMotionUpdatesToQueue:[NSOperationQueue mainQueue]
withHandler:^(CMDeviceMotion *deviceMotion, NSError *error) {

         //Access to all the data…
          deviceMotion.attitude,
          deviceMotion.rotationRate,
          deviceMotion.gravity,
          deviceMotion.userAcceleration,
          deviceMotion.magneticField,


     }];
 }
```

declare

instantiate

if device is capable

queue to run on

how often to push updates

getting updates

```
// for getting access to the fused motion data (best practice, filtered)
@property (nonatomic,strong) CMMotionManager *mManager;



self.mManager = [[CMMotionManager alloc] init];

 if([self.mManager isDeviceMotionAvailable])
 {

        [self.mManager setDeviceMotionUpdateInterval:yourSamplingIntervalInSeconds];
        [self.mManager startDeviceMotionUpdatesToQueue:[NSOperationQueue mainQueue]
withHandler:^(CMDeviceMotion *deviceMotion, NSError *error) {

           //Access to all the data…
            deviceMotion.attitude,
            deviceMotion.rotationRate,
            deviceMotion.gravity,
            deviceMotion.userAcceleration,
            deviceMotion.magneticField,


        }];
    }
```

declare

instantiate

if device is capable

queue to run on

how often to push updates

the data

getting updates

```
// for getting access to the fused motion data (best practice, filtered)
@property (nonatomic,strong) CMMotionManager *mManager;



self.mManager = [[CMMotionManager alloc] init];

 if([self.mManager isDeviceMotionAvailable])
 {

        [self.mManager setDeviceMotionUpdateInterval:yourSamplingIntervalInSeconds];
        [self.mManager startDeviceMotionUpdatesToQueue:[NSOperationQueue mainQueue]
withHandler:^(CMDeviceMotion *deviceMotion, NSError *error) {

            //Access to all the data…
             deviceMotion.attitude,
             deviceMotion.rotationR

        }];
    }
```

declare

instantiate

if device is capable

how often to push updates

queue to run on

the data

device motion demo

device motion demo

- lets build something

device motion demo

- lets build something
  - to start: the motion plotter

device motion demo

- lets build something
  - to start: the motion plotter
  - custom views and a simple grapher from Apple

device motion demo

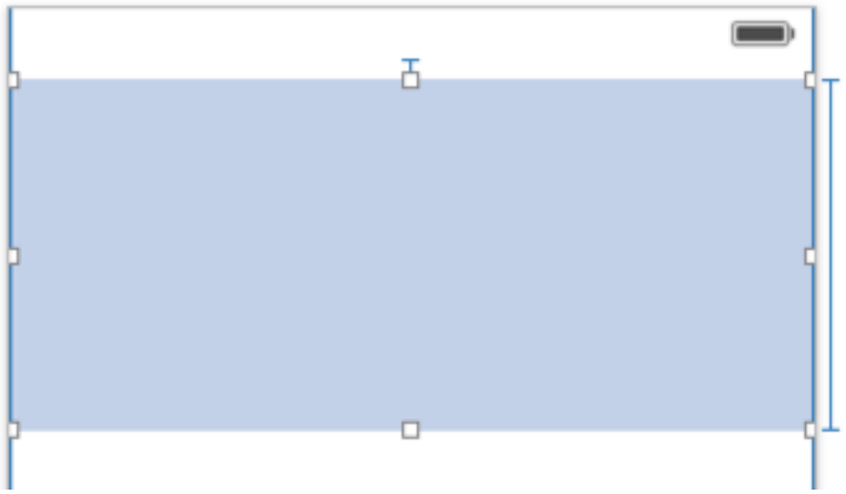
- lets build something
  - to start: the motion plotter
  - custom views and a simple grapher from Apple

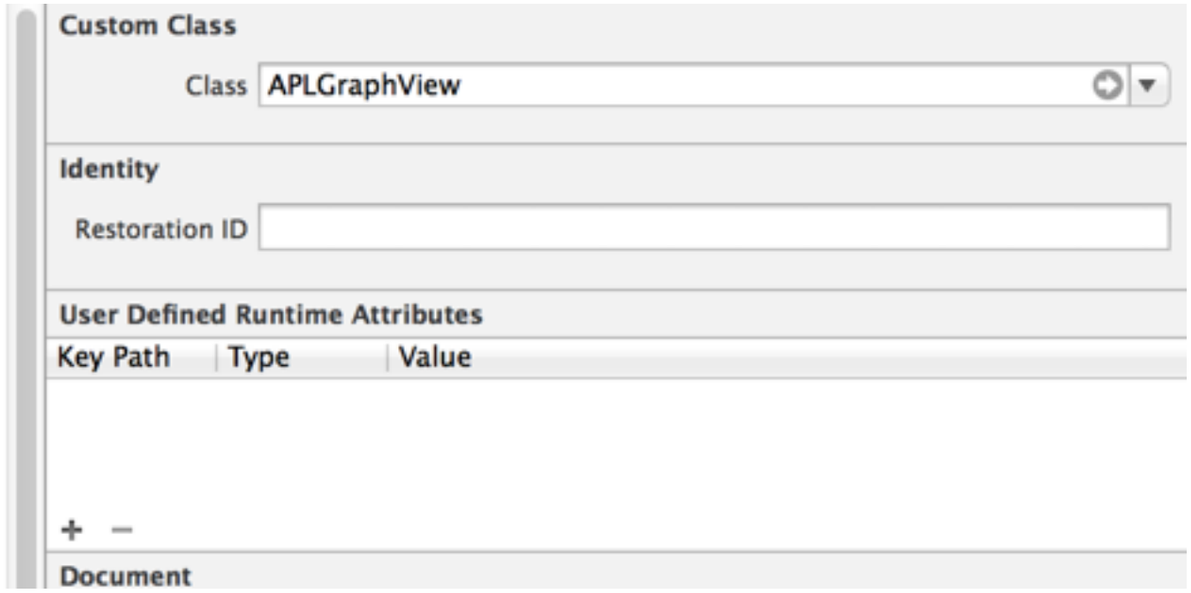
```
@interface APLGraphView : UIView
-(void)addX:(double)x y:(double)y z:(double)z;
@end
```

device motion demo

- lets build something
  - to start: the motion plotter
  - custom views and a simple grapher from Apple

```
@interface APLGraphView : UIView
-(void)addX:(double)x y:(double)y z:(double)z;
@end
```





phone trajectory

phone trajectory

- what direction is the phone (user) headed?
- direction could be:
  - cardinal {N, S, E, W}
  - altitude {sea level, +30 feet, etc.}
  - relative altitude {up, down}
  - relative trajectory {left, right, straight}

phone trajectory

- what direction is the phone (user) headed?
- direction could be:
  - cardinal {N, S, E, W}
  - altitude {sea level, +30 feet, etc.}
  - relative altitude {up, down}
  - relative trajectory {left, right, straight}
- how should we sense each of these?

phone trajectory

- what direction is the phone (user) headed?
- direction could be:
  - cardinal {N, S, E, W}
  - altitude {sea level, +30 feet, etc.}
  - relative altitude {up, down}
  - relative trajectory {left, right, straight}
- how should we sense each of these?

GPS and magnetometer

phone trajectory

- what direction is the phone (user) headed?
- direction could be:
  - cardinal {N, S, E, W}
  - altitude {sea level, +30 feet, etc.}
  - relative altitude {up, down}
  - relative trajectory {left, right, straight}
- how should we sense each of these?

GPS and magnetometer

GPS

phone trajectory

- what direction is the phone (user) headed?
- direction could be:
  - cardinal {N, S, E, W}
  - altitude {sea level, +30 feet, etc.}
  - relative altitude {up, down}
  - relative trajectory {left, right, straight}
- how should we sense each of these?

GPS and magnetometer

GPS

motion sensors

phone trajectory

- what direction is the phone (user) headed?
- direction could be:
  - cardinal {N, S, E, W}
  - altitude {sea level, +30 feet, etc.}
  - relative altitude {up, down}
  - relative trajectory {left, right, straight}
- how should we sense each of these?

GPS and magnetometer

GPS

motion sensors

motion sensors

up/down movement

- questions:
  - are we accelerating?
  - in what direction are we accelerating?
  - are we accelerating opposite of gravity?

up/down movement

- questions:
  - are we accelerating?
  - in what direction are we accelerating?
  - are we accelerating opposite of gravity?

which way is gravity?

up/down movement

- questions:
  - are we accelerating?
  - in what direction are we accelerating?
  - are we accelerating opposite of gravity?

which way is gravity?

```
deviceMotion.gravity.{x,y,z}
```

up/down movement

- questions:
  - are we accelerating?
  - in what direction are we accelerating?
  - are we accelerating opposite of gravity?

which way is gravity?

```
deviceMotion.gravity.{x,y,z}
```

which way is the phone accelerating?

up/down movement

- questions:
  - are we accelerating?
  - in what direction are we accelerating?
  - are we accelerating opposite of gravity?

which way is gravity?

```
deviceMotion.gravity.{x,y,z}
```

which way is the phone accelerating?

```
deviceMotion.userAcceleration.{x,y,z}
```

up/down movement

- questions:
  - are we accelerating?
  - in what direction are we accelerating?
  - are we accelerating opposite of gravity?

which way is gravity?

```
deviceMotion.gravity.{x,y,z}
```

which way is the phone accelerating?

```
deviceMotion.userAcceleration.{x,y,z}
```

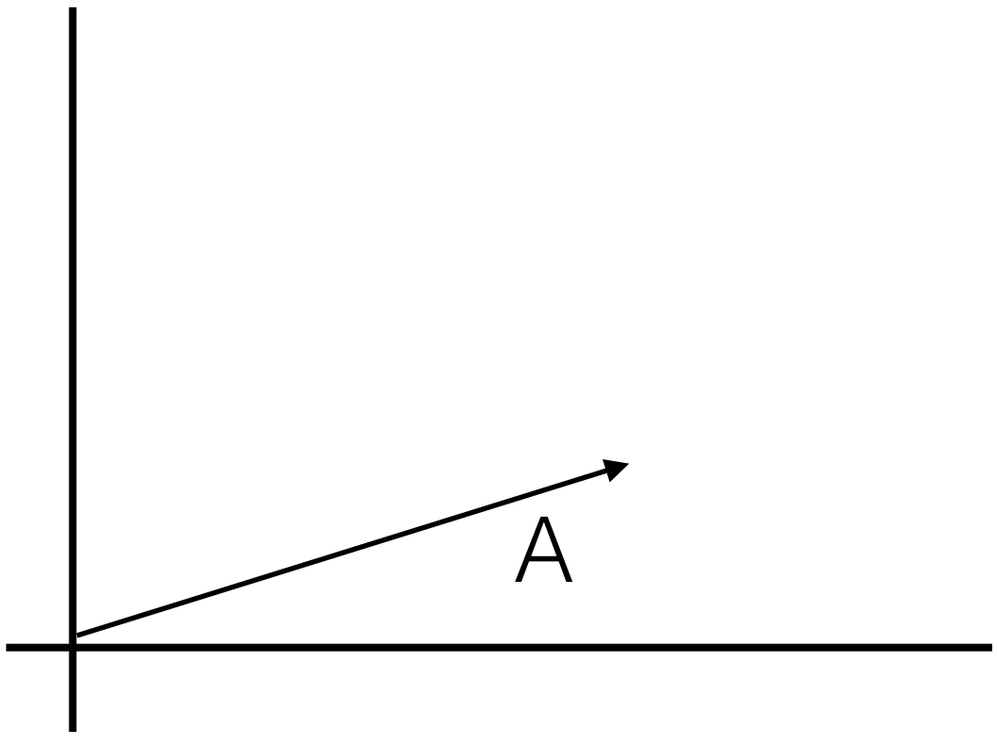
vectors

vector direction

- how much of one vector is in the direction of another?
- projections

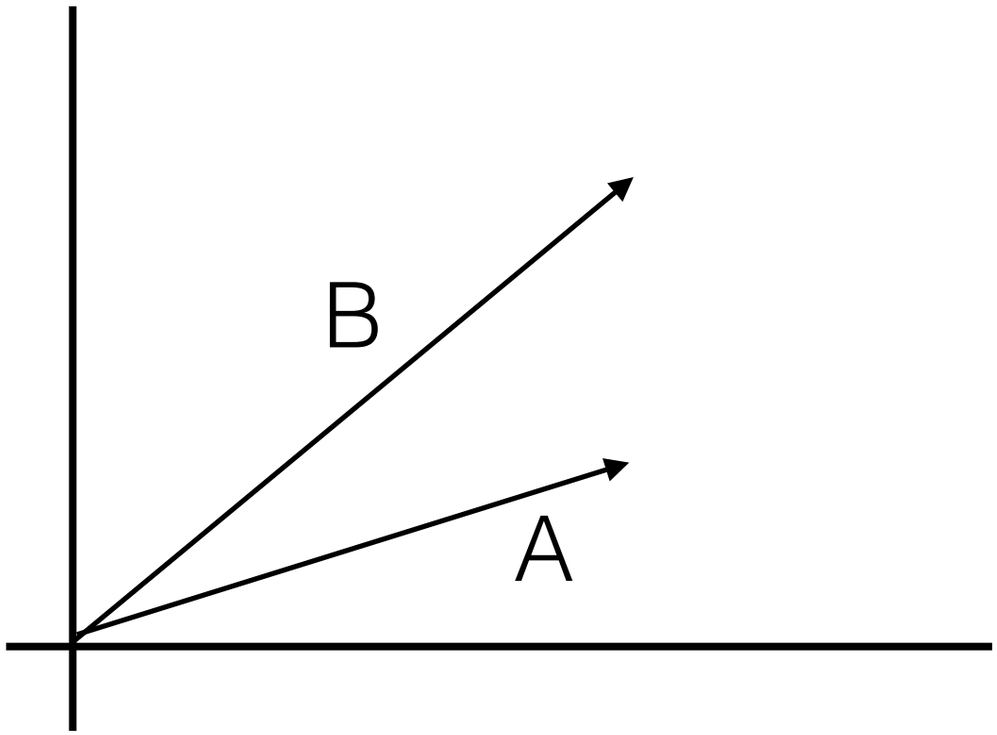
vector direction

- how much of one vector is in the direction of another?
- projections



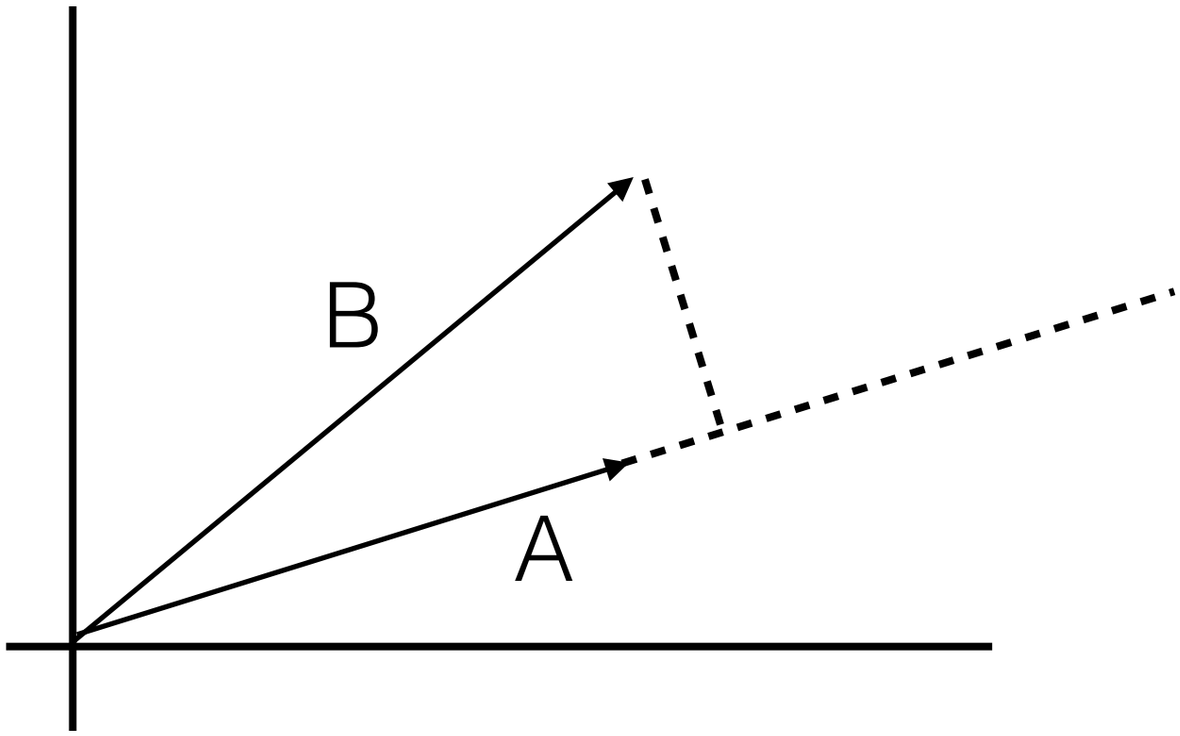
vector direction

- how much of one vector is in the direction of another?
- projections



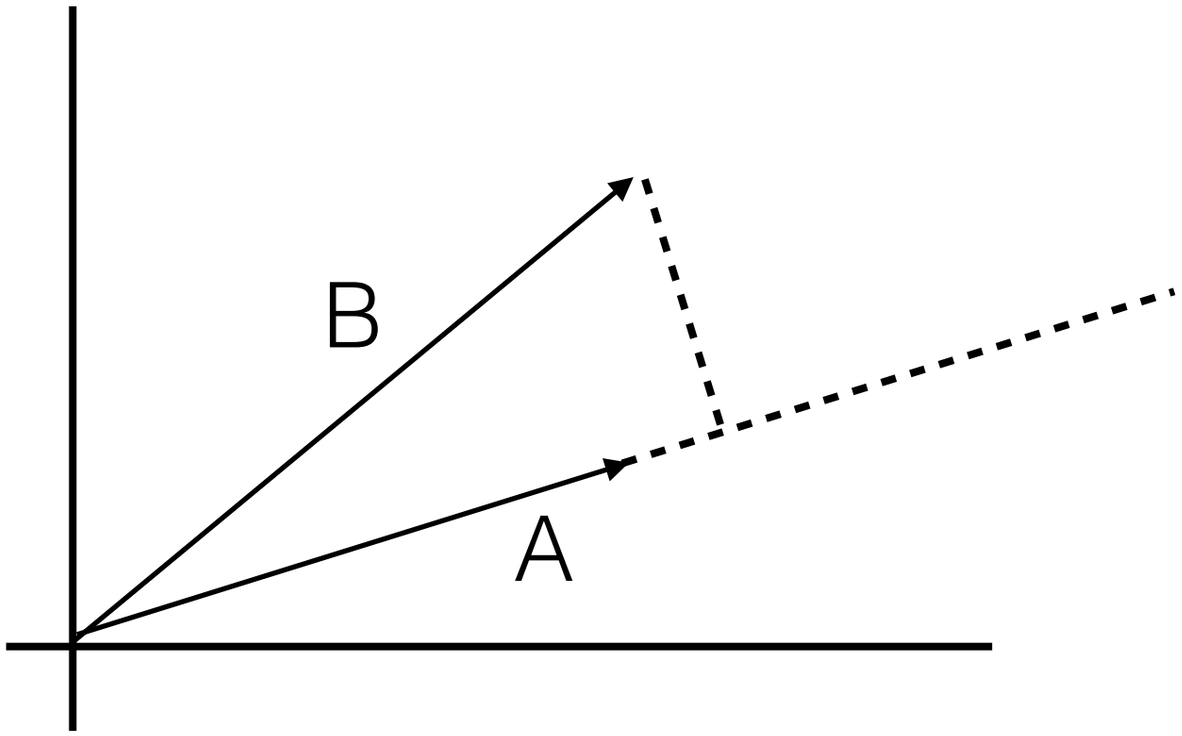
vector direction

- how much of one vector is in the direction of another?
- projections



vector direction

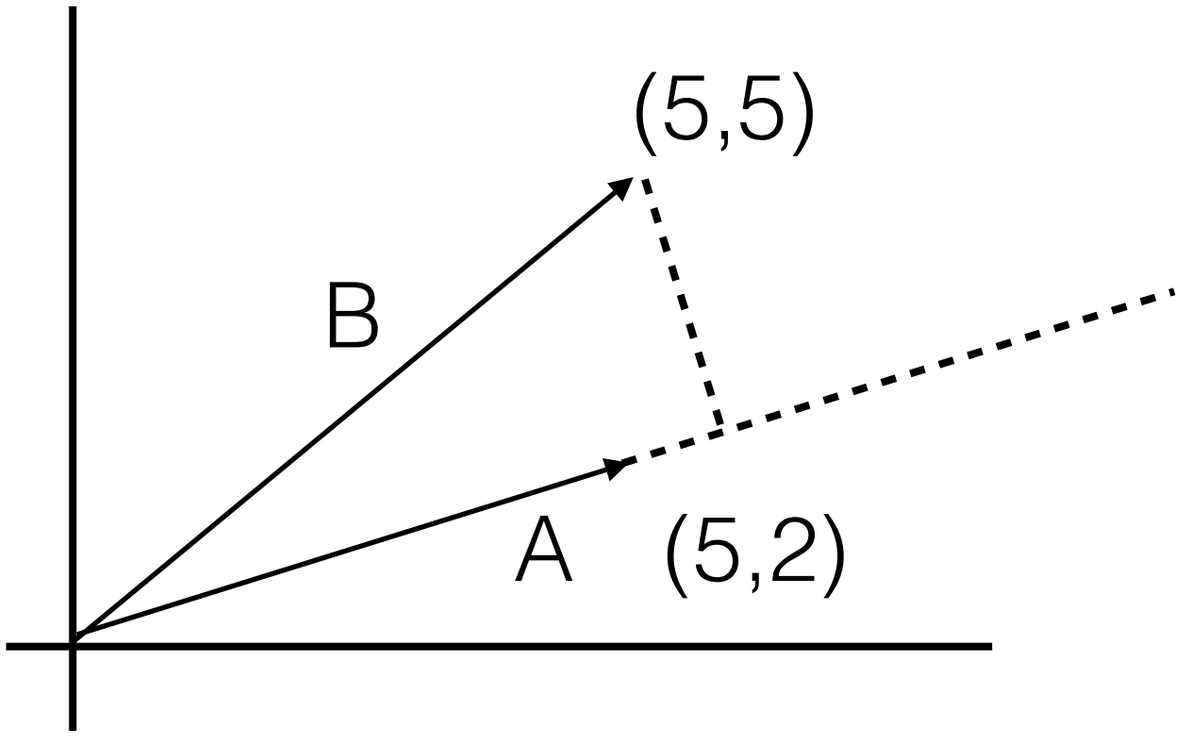
- how much of one vector is in the direction of another?
- projections



$$\frac{A \bullet B}{|A|}$$

vector direction

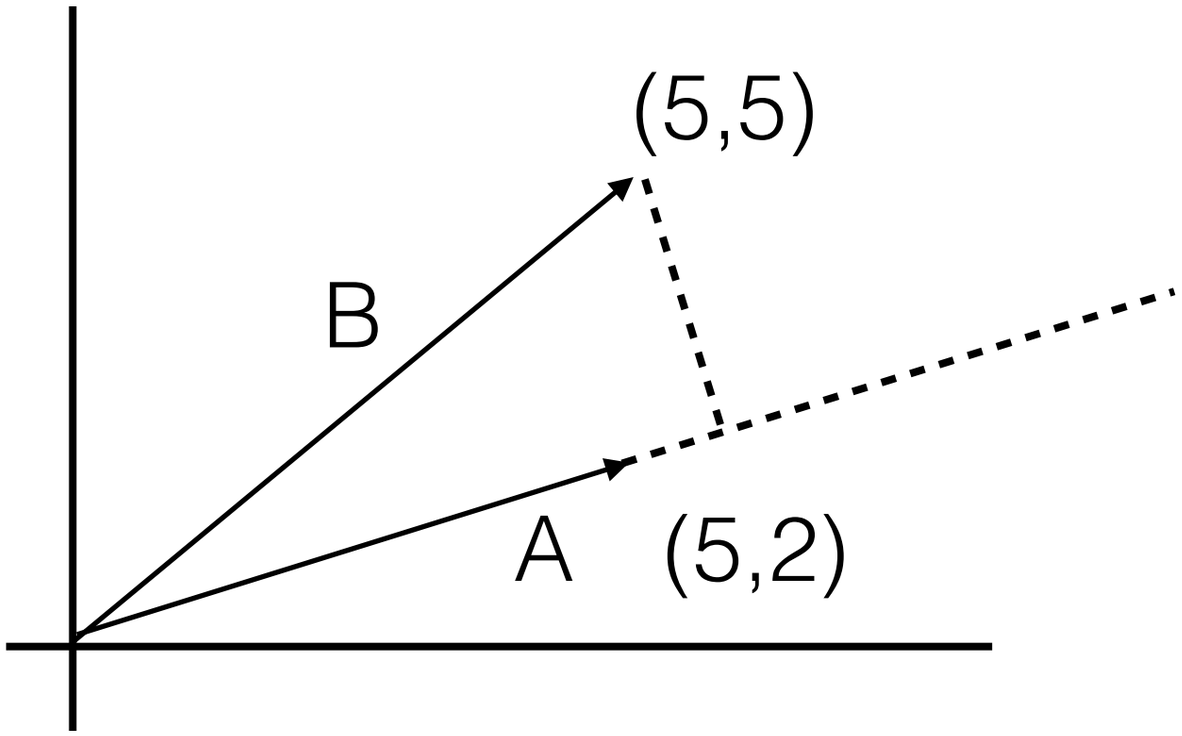
- how much of one vector is in the direction of another?
- projections



$$\frac{A \bullet B}{|A|}$$

vector direction

- how much of one vector is in the direction of another?
- projections

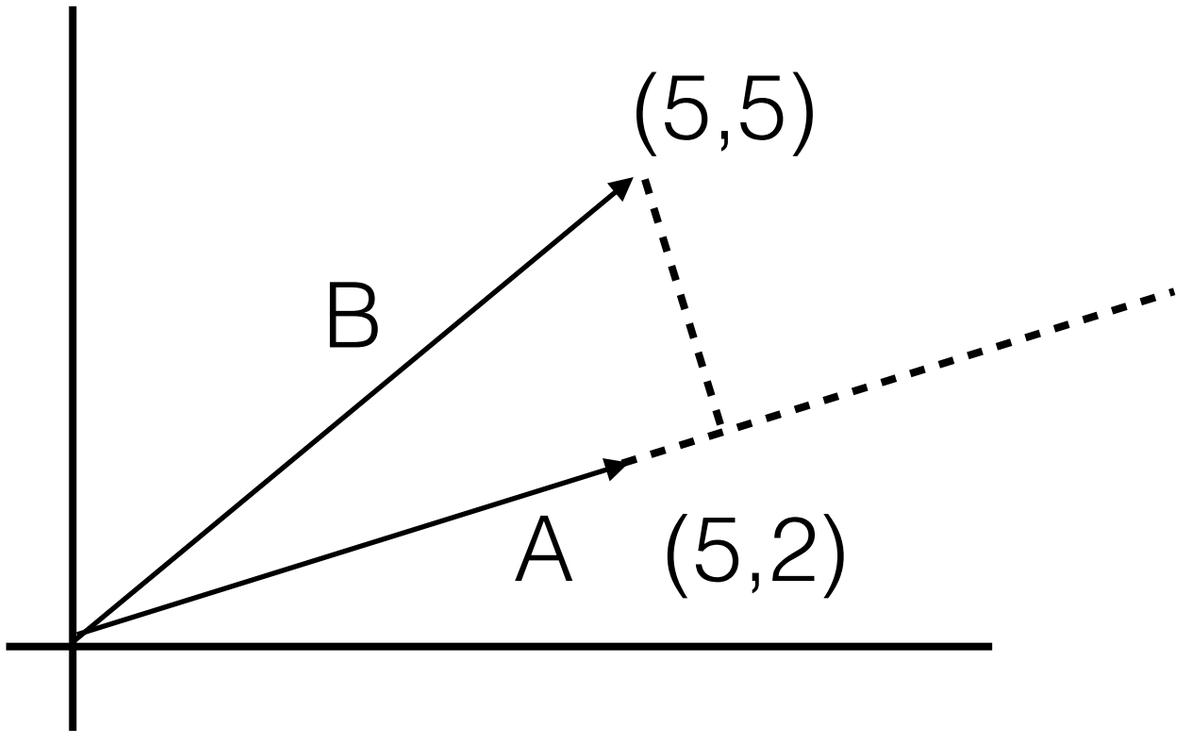


$$\frac{A \bullet B}{|A|}$$

$$\frac{(5,5)\bullet(5,2)}{|(5,2)|}$$

vector direction

- how much of one vector is in the direction of another?
- projections



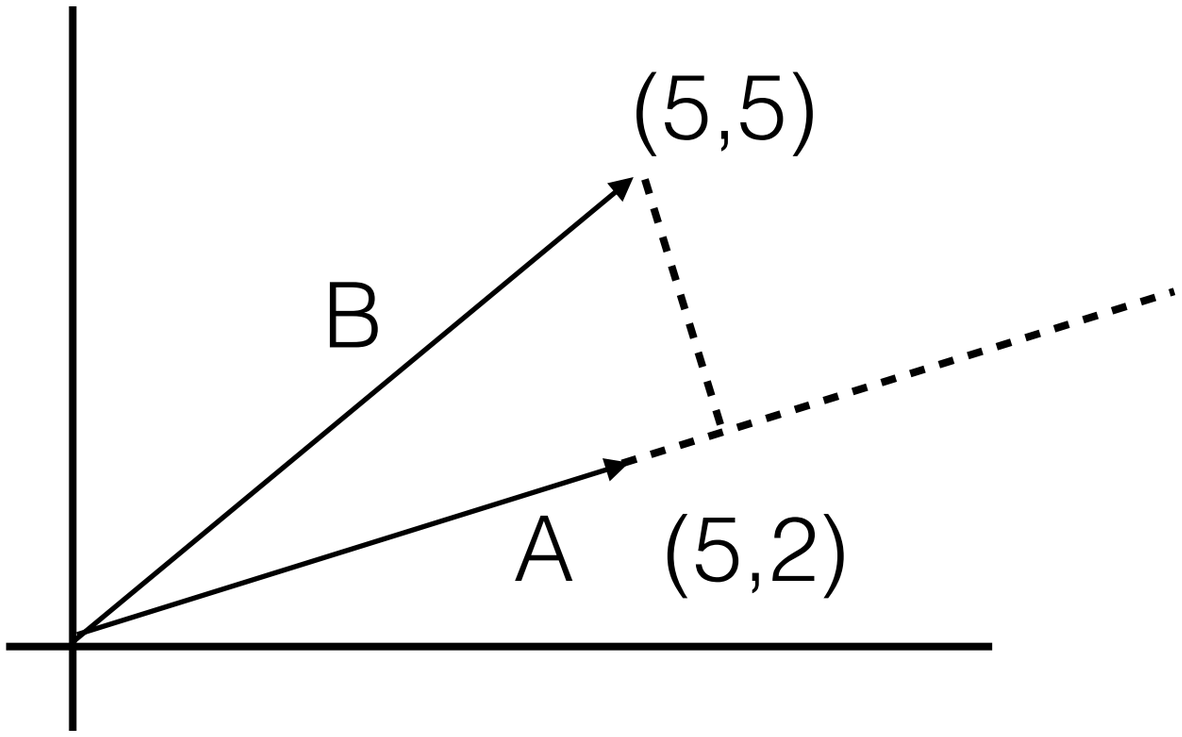
$$\frac{A \bullet B}{|A|}$$

$$\frac{(5,5)\bullet(5,2)}{|(5,2)|}$$

$$\frac{5*5+5*2}{\sqrt{(5^2+2^2)}}$$

vector direction

- how much of one vector is in the direction of another?
- projections



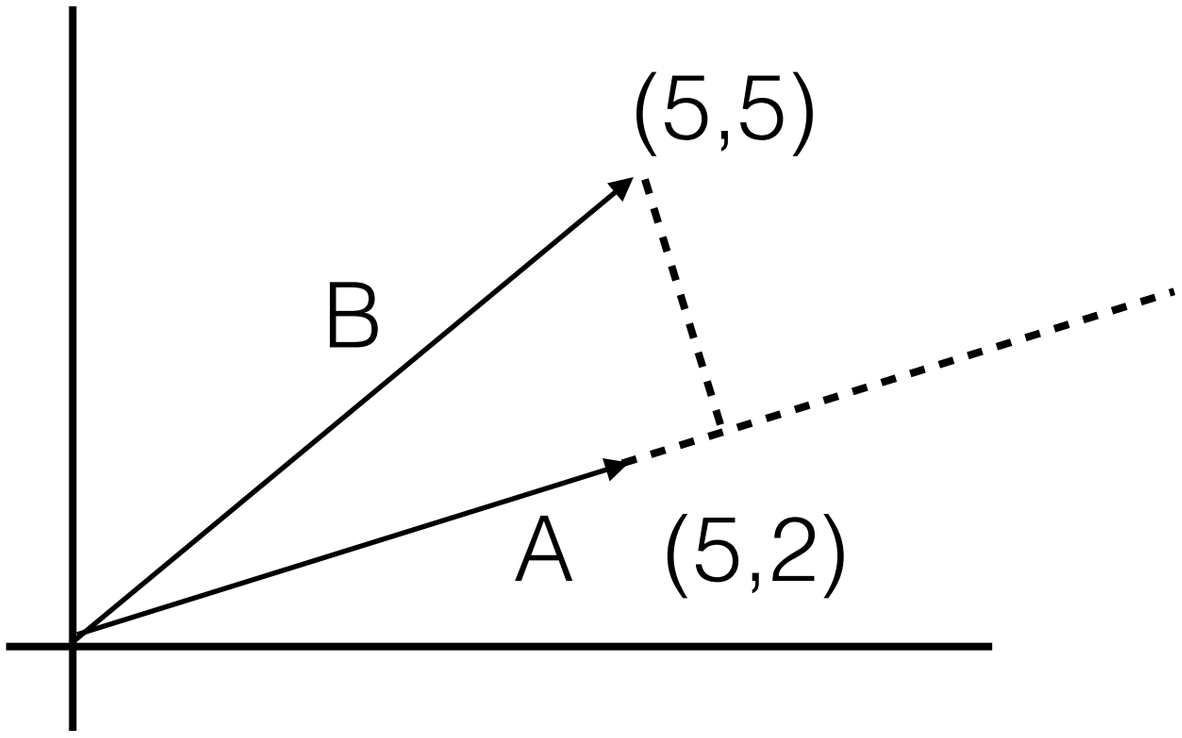
$$\frac{A \bullet B}{|A|}$$

$$\frac{(5,5)\bullet(5,2)}{|(5,2)|}$$

$$\frac{5*5+5*2}{\sqrt{(5^2+2^2)}} = 35/\sqrt{29}$$

vector direction

- how much of one vector is in the direction of another?
- projections



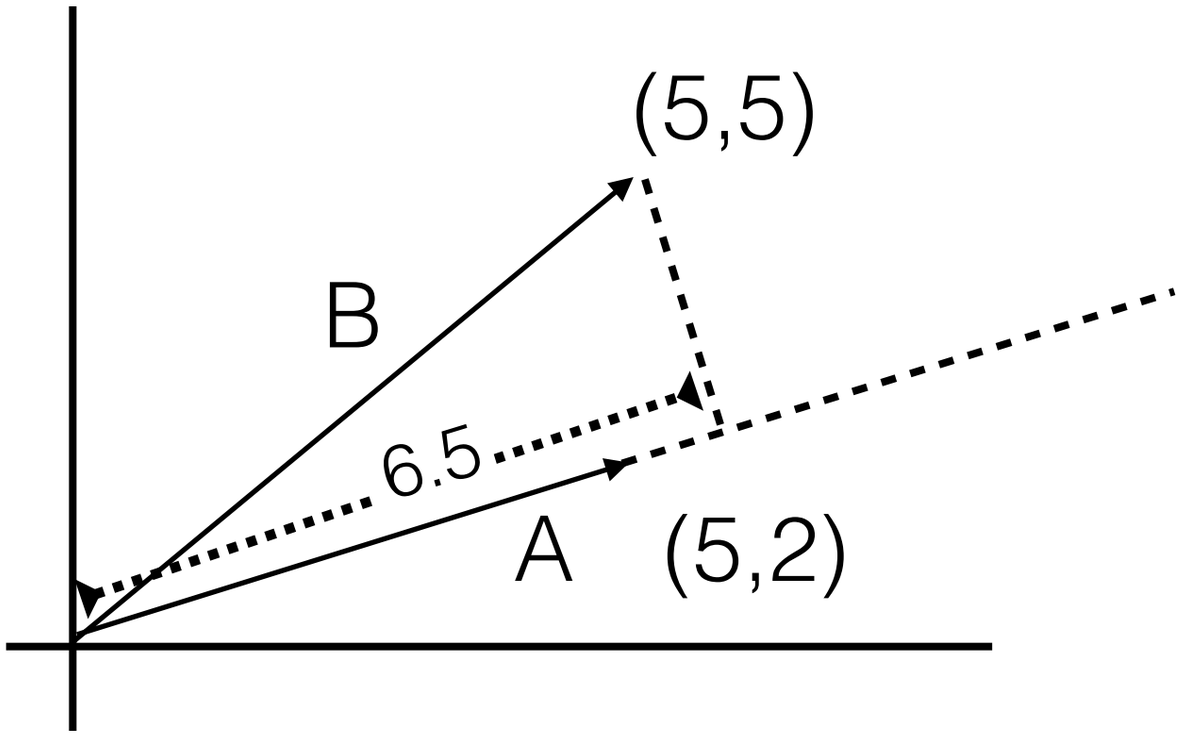
$$\frac{A \bullet B}{|A|}$$

$$\frac{(5,5)\bullet(5,2)}{|(5,2)|}$$

$$\frac{5*5+5*2}{\sqrt{(5^2+2^2)}} = 35/\sqrt{29} \sim 6.5$$

vector direction

- how much of one vector is in the direction of another?
- projections



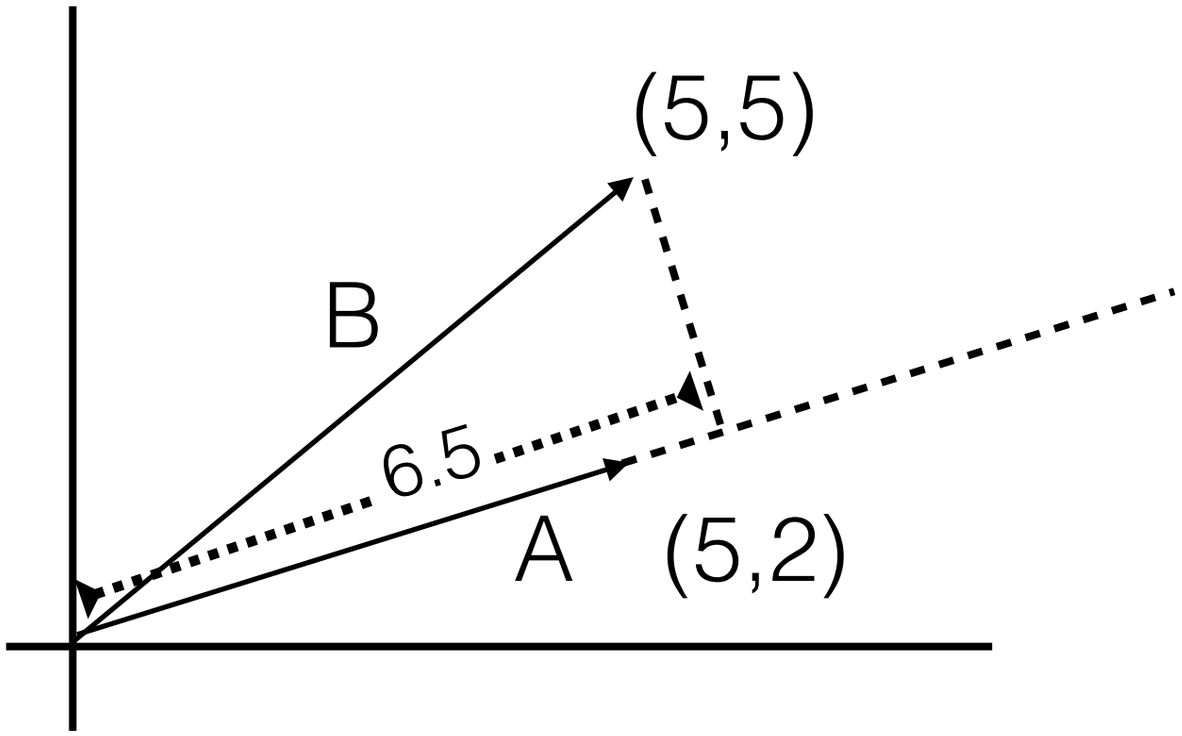
$$\frac{A \bullet B}{|A|}$$

$$\frac{(5,5)\bullet(5,2)}{|(5,2)|}$$

$$\frac{5*5+5*2}{\sqrt{(5^2+2^2)}} = 35/\sqrt{29} \sim 6.5$$

vector direction

- how much of one vector is in the direction of another?
- projections



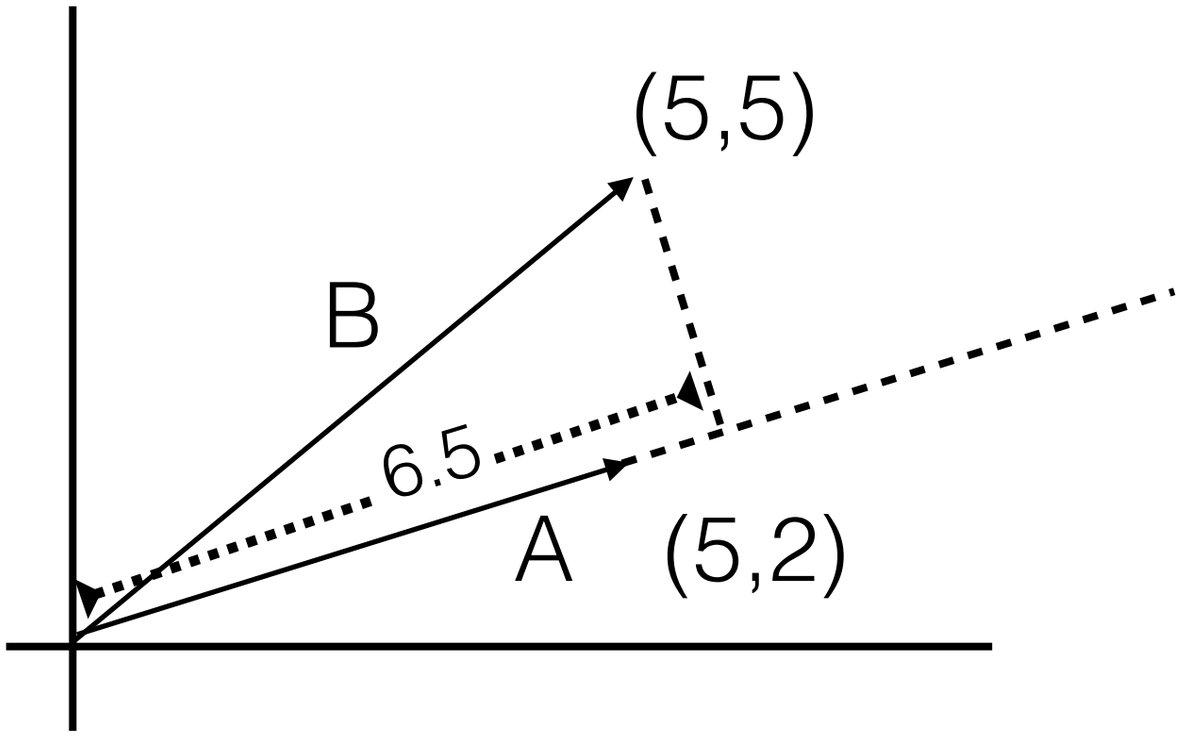
$$\frac{A \bullet B}{|A|}$$

$$\frac{(5,5)\bullet(5,2)}{|(5,2)|}$$

$$\frac{5*5+5*2}{\sqrt{(5^2+2^2)}} = 35/\sqrt{29} \sim 6.5$$

vector direction

- how much of one vector is in the direction of another?
- projections



$$\frac{A \bullet B}{|A|}$$

$$\frac{(5,5)\bullet(5,2)}{|(5,2)|}$$

$$\frac{5*5+5*2}{\sqrt{(5^2+2^2)}} = 35/\sqrt{29} \sim 6.5$$

vector direction

- acceleration of the user towards or away from gravity?

vector direction

- acceleration of the user towards or away from gravity?

```
CMAcceleration gravity, CMAcceleration userAccel

float dotProduct =
  gravity.x*userAccel.x + gravity.y*userAccel.y + gravity.z*userAccel.z;

float normDotProd =
  dotProduct / (gravity.x*gravity.x + gravity.y*gravity.y + gravity.z*gravity.z);
```

vector direction

- acceleration of the user towards or away from gravity?

```
CMAcceleration gravity, CMAcceleration userAccel

float dotProduct =
  gravity.x*userAccel.x + gravity.y*userAccel.y + gravity.z*userAccel.z;

float normDotProd =
  dotProduct / (gravity.x*gravity.x + gravity.y*gravity.y + gravity.z*gravity.z);
```

positive acceleration is speeding up
negative acceleration is slowing down

vector acceleration demo

- don’t drop it!

the end of motion…

- before moving on…
- how might you perform step climbing?
- how might you debug your algorithm?

what is image processing

- the art and science of manipulating pixels
  - combining images (blending or compositing)
  - enhancing edges and lines
  - adjusting contrast, color
  - warping, transformation
  - filtering
  - …anything you can do in photoshop
  - also used in computer vision

what is image processing

- the art and science of manipulating pixels

images as data

images as data

- an image can be represented in many ways

images as data

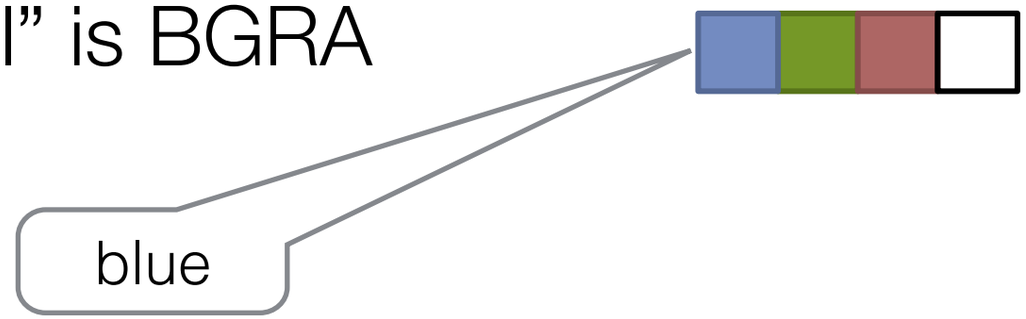
- an image can be represented in many ways
- most common format is a matrix of pixels
  - each “pixel” is BGRA

images as data

- an image can be represented in many ways
- most common format is a matrix of pixels
  - each “pixel” is BGRA

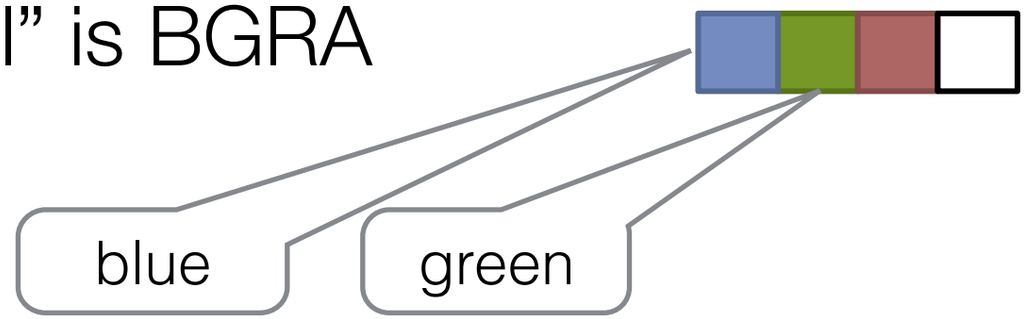
images as data

- an image can be represented in many ways
- most common format is a matrix of pixels
  - each “pixel” is BGRA



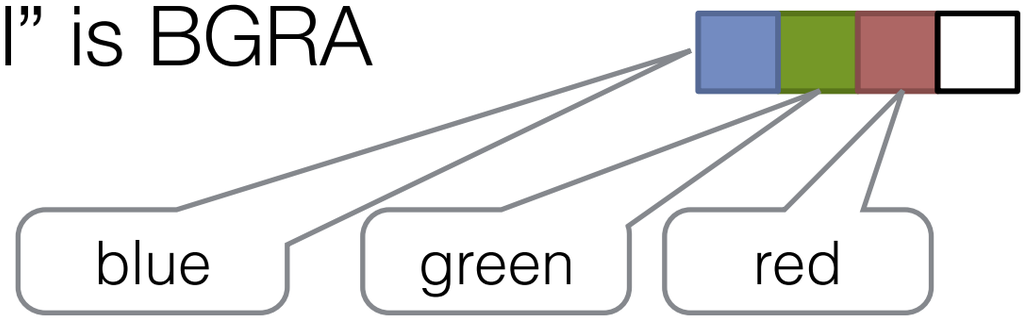
images as data

- an image can be represented in many ways
- most common format is a matrix of pixels
  - each “pixel” is BGRA



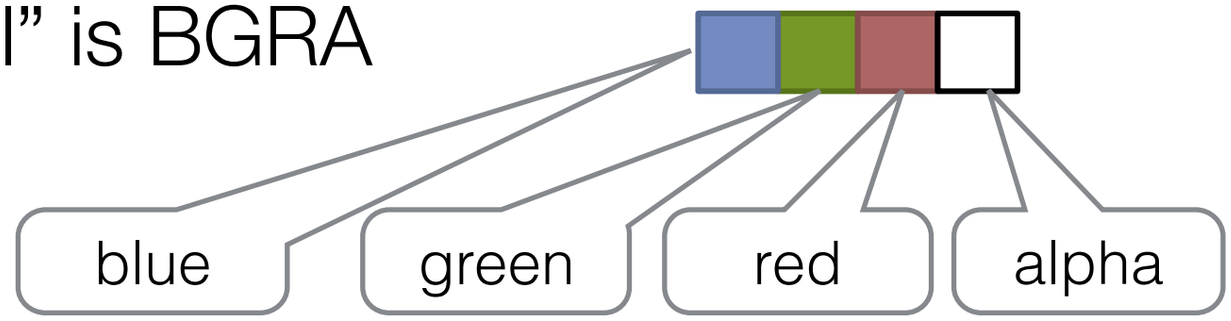
images as data

- an image can be represented in many ways
- most common format is a matrix of pixels
  - each “pixel” is BGRA



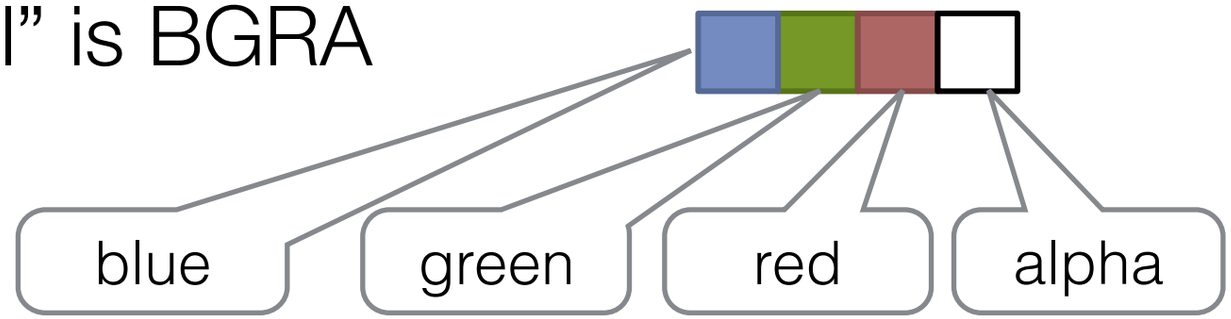
images as data

- an image can be represented in many ways
- most common format is a matrix of pixels
  - each “pixel” is BGRA



images as data

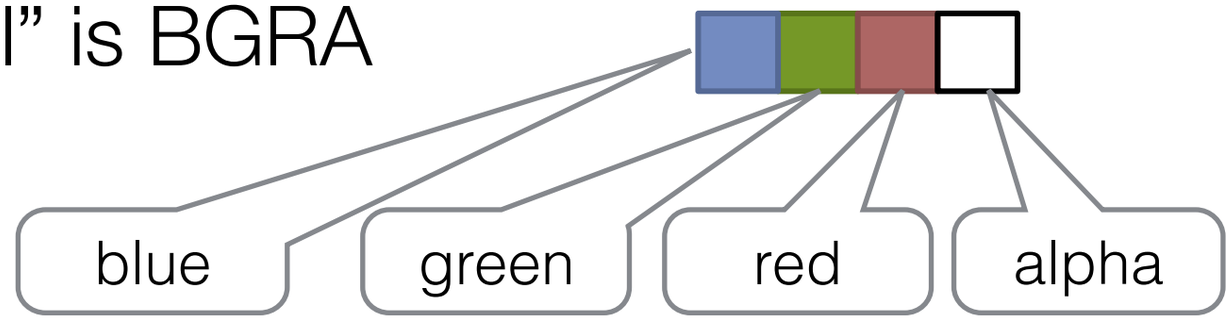
- an image can be represented in many ways
- most common format is a matrix of pixels
  - each “pixel” is BGRA



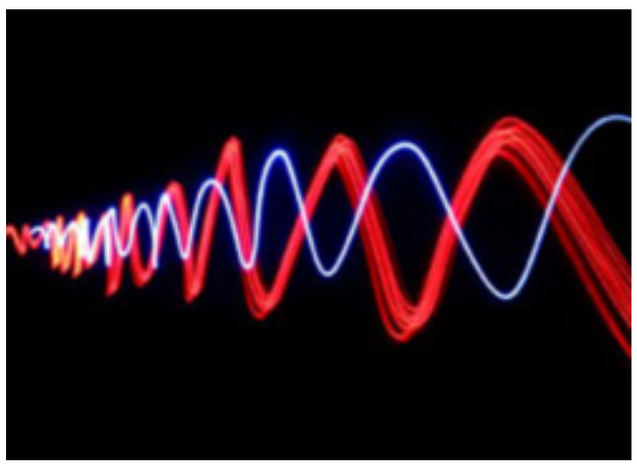
- used for capture and display

images as data

- an image can be represented in many ways
- most common format is a matrix of pixels
  - each “pixel” is BGRA

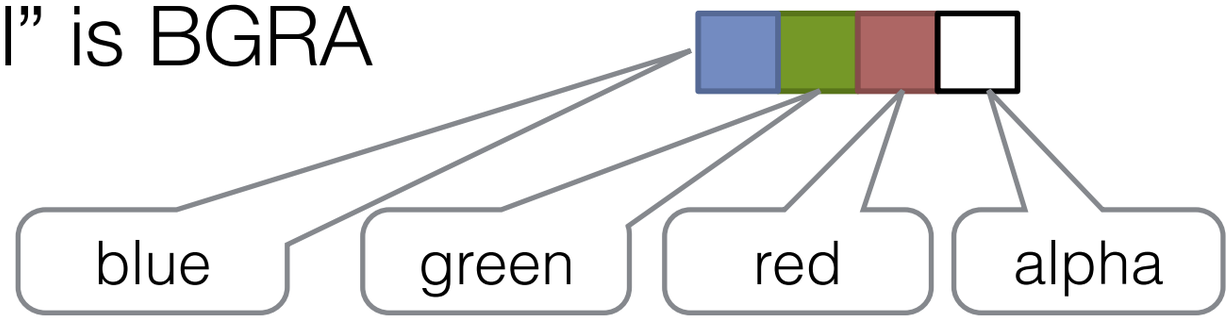


- used for capture and display

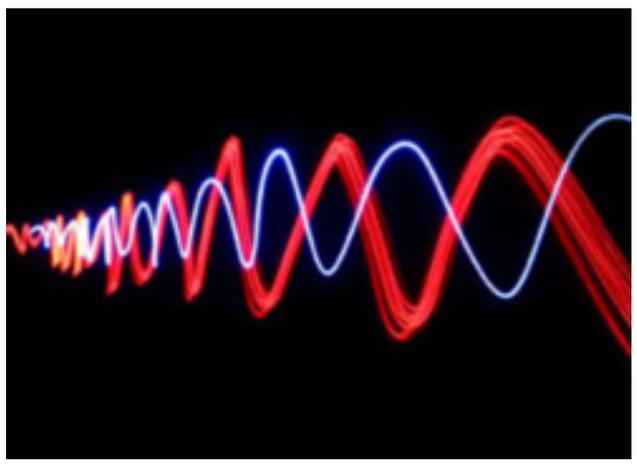


images as data

- an image can be represented in many ways
- most common format is a matrix of pixels
  - each “pixel” is BGRA



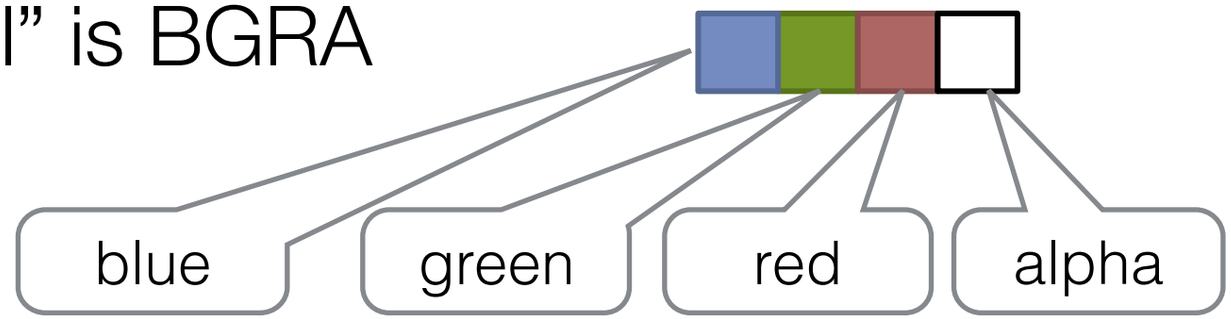
- used for capture and display



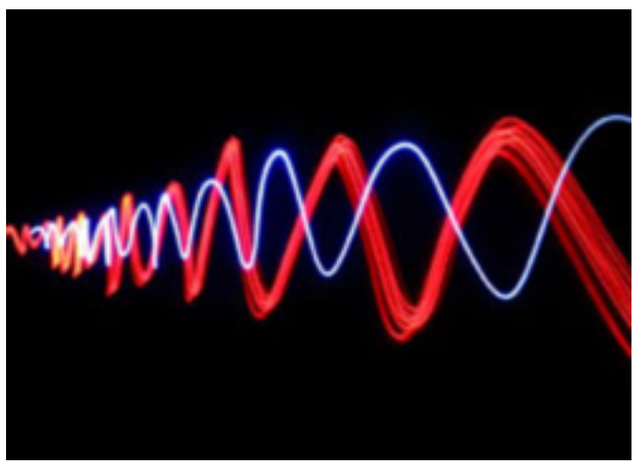
sensor

images as data

- an image can be represented in many ways
- most common format is a matrix of pixels
  - each “pixel” is BGRA



- used for capture and display

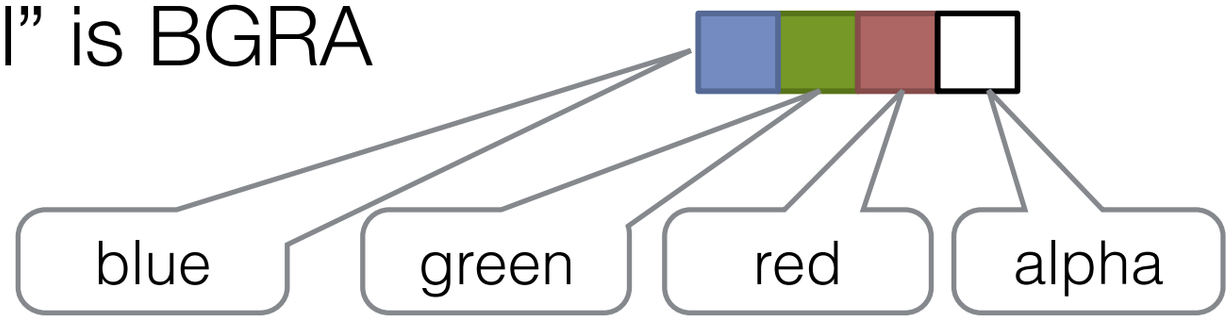


sensor

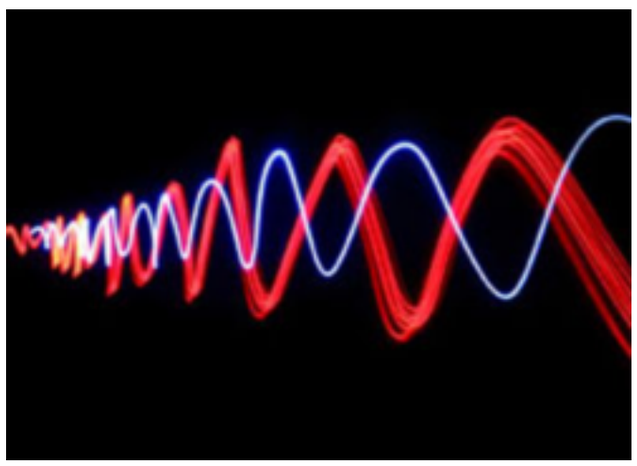
sensor

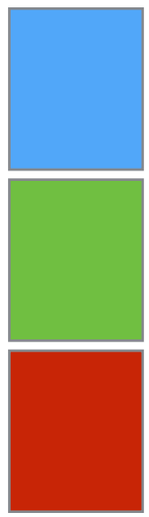
images as data

- an image can be represented in many ways
- most common format is a matrix of pixels
  - each “pixel” is BGRA



- used for capture and display





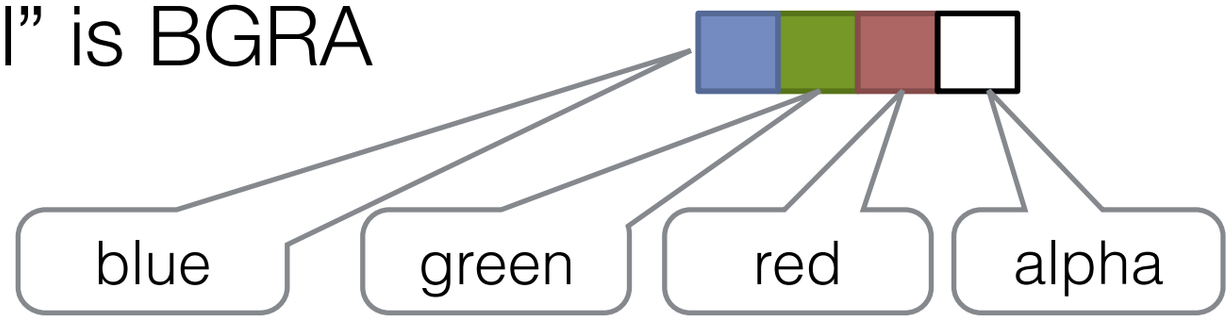
sensor

sensor

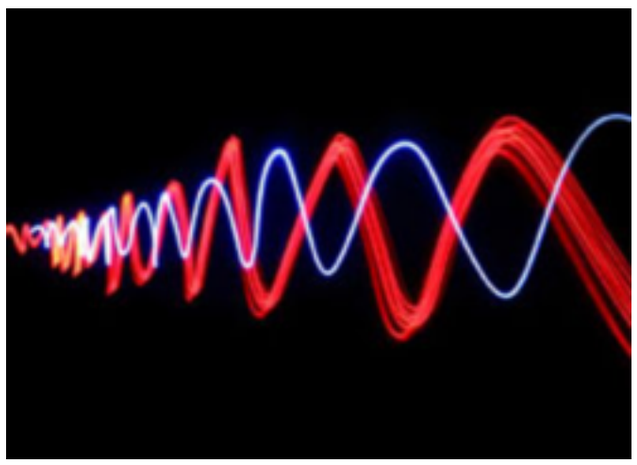
sensor

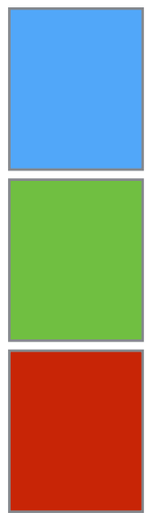
images as data

- an image can be represented in many ways
- most common format is a matrix of pixels
  - each “pixel” is BGRA



- used for capture and display

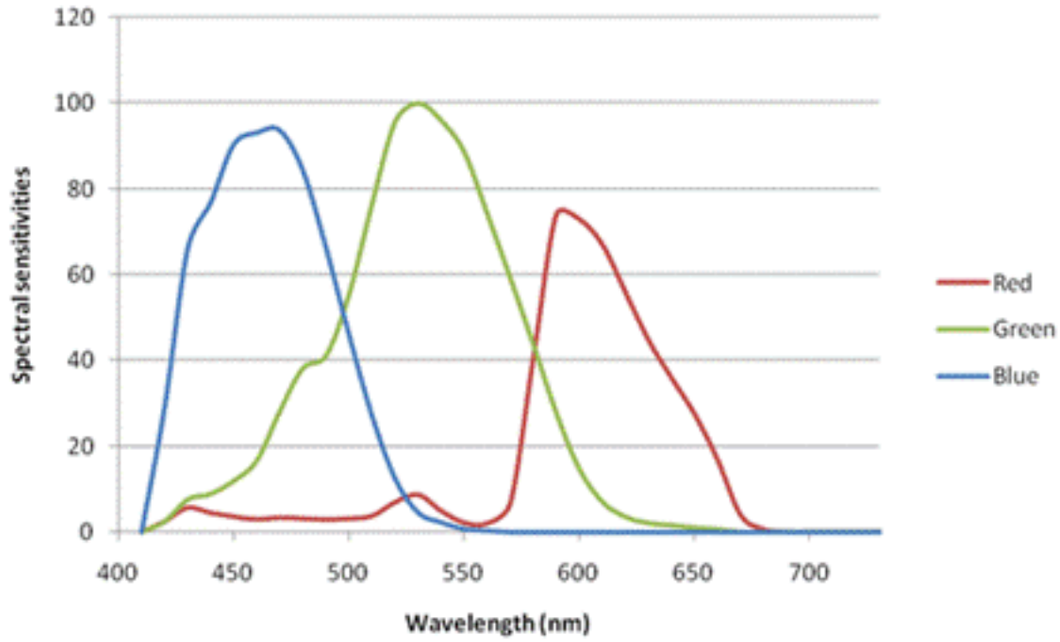




sensor

sensor

sensor

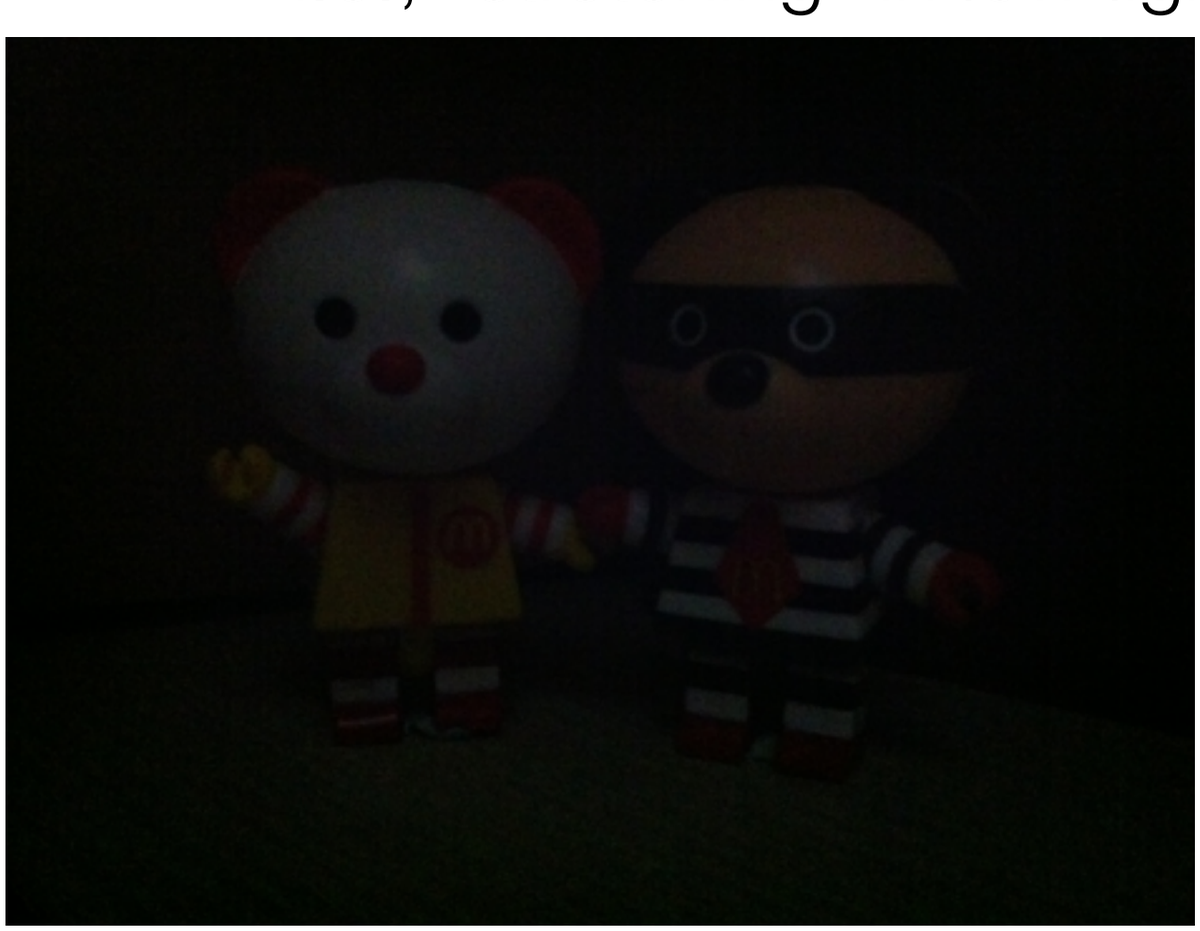


images as signals

- everything from audio still applies
- quantization
  - each pixel can only take on 0-255 colors
  - i.e., “stretching” in low light conditions

images as signals

- everything from audio still applies
- quantization
  - each pixel can only take on 0-255 colors
  - i.e., “stretching” in low light conditions

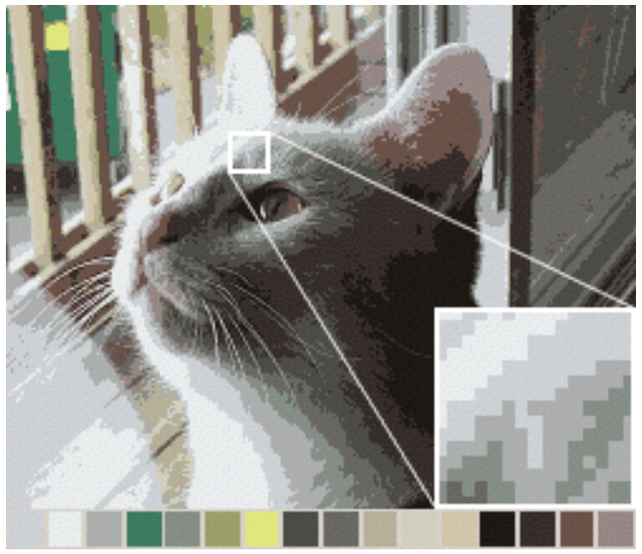


images as signals

- everything from audio still applies
- quantization
  - each pixel can only take on 0-255 colors
  - i.e., “stretching” in low light conditions

images as signals

- everything from audio still applies
- quantization
  - each pixel can only take on 0-255 colors
  - i.e., “stretching” in low light conditions



sampling errors

sampling errors

- in time (video)

sampling errors

- in time (video)

sampling errors

- in time (video)

- in space (resolution)
  - “frequency” is in terms of pixels

sampling errors

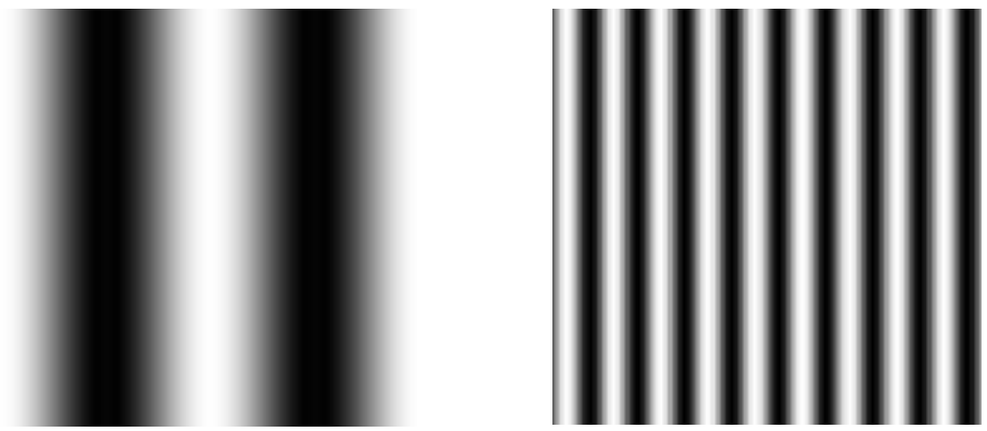
- in time (video)

- in space (resolution)
  - “frequency” is in terms of pixels

sampling errors

- in time (video)

- in space (resolution)
  - “frequency” is in terms of pixels



images as signals

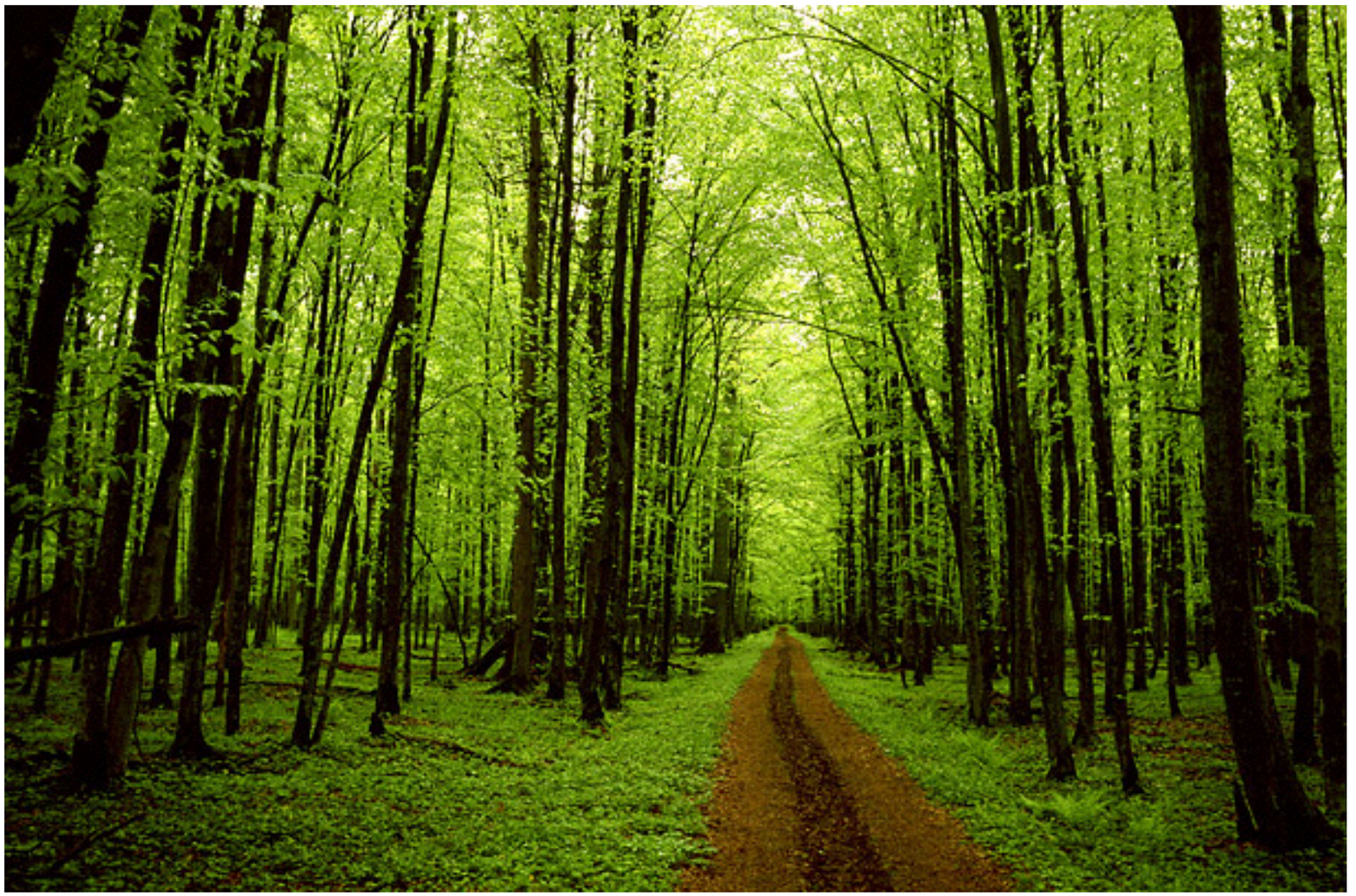


image with lots of high frequency

images as signals

image with lots of low frequency

what is filtering?

- same as audio
  - convolution (linear)

.11	.11	.11
.11	.11	.11
.11	.11	.11

kernel

image

1	4	2	5	6	9
1	4	2	5	5	9
1	4	2	8	8	7
3	4	3	9	9	8
1	0	2	7	7	9
1	4	3	9	8	6
2	4	2	8	7	9

what is filtering?

- same as audio
  - convolution (linear)

kernel

.11	.11	.11	image			
.11	.11	.11	2	5	6	9
.11	.11	.11	2	5	5	9
	1	4	2	8	8	7
3	4	3	9	9	8	
1	0	2	7	7	9	
1	4	3	9	8	6	
2	4	2	8	7	9	

what is filtering?

- same as audio
  - convolution (linear)

image

				.11	.11	.11
1.1	1.5	2.4	2.7	.13	.12	.11
1	4	2	5	.51	.91	.11
1	4	2	8	8	7	
3	4	3	9	9	8	
1	0	2	7	7	9	
1	4	3	9	8	6	
2	4	2	8	7	9	

kernel

what is filtering?

- same as audio
  - convolution (linear)

image

1.1	1.5	2.4	2.7	4.3	3.2
1.6	2.3	4.0	4.7	6.8	4.8
1.8	2.6	4.5	5.6	7.5	5.1
1.4	2.2	4.3	6.1	8.0	5.2
1.4	2.3	4.5	6.3	8.0	5.2
1.3	2.1	4.3	5.8	7.7	5.1
1.2	1.7	3.3	4.1	5.2	3.3

kernel

.11	.11	.11
.11	.11	.11
.11	.11	.11

what is filtering?

-1	0	1
-1	0	1
-1	0	1

1	4	2	5	6	9
1	4	2	5	5	9
1	4	2	8	8	7
3	4	3	9	9	8
1	0	2	7	7	9
1	4	3	9	8	6
2	4	2	8	7	9

what is filtering?

-1	0	1
-1	0	1
-1	0	1

-8	-2	-2	-7	-8	4
-12	-3	-6	-13	-7	5
-8	-2	-10	-15	-2	2
-8	-2	-16	-17	0	3
-8	-3	-17	-16	2	4
-8	-3	-16	-15	0	1
-8	-2	-9	-10	2	2

interesting operations

- the power of simple filtering
  - mathworks demo

interesting operations

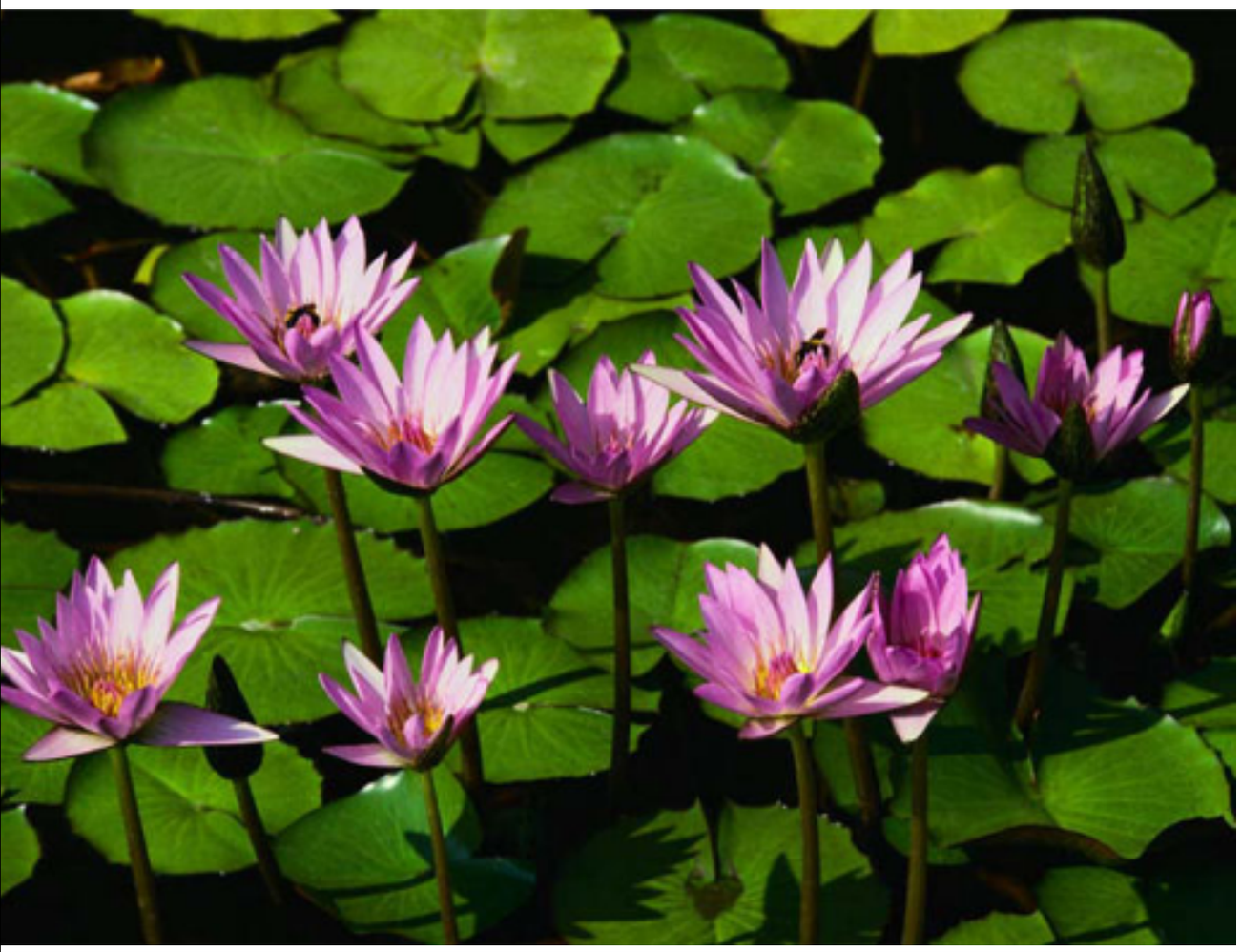
- the power of simple filtering
  - mathworks demo

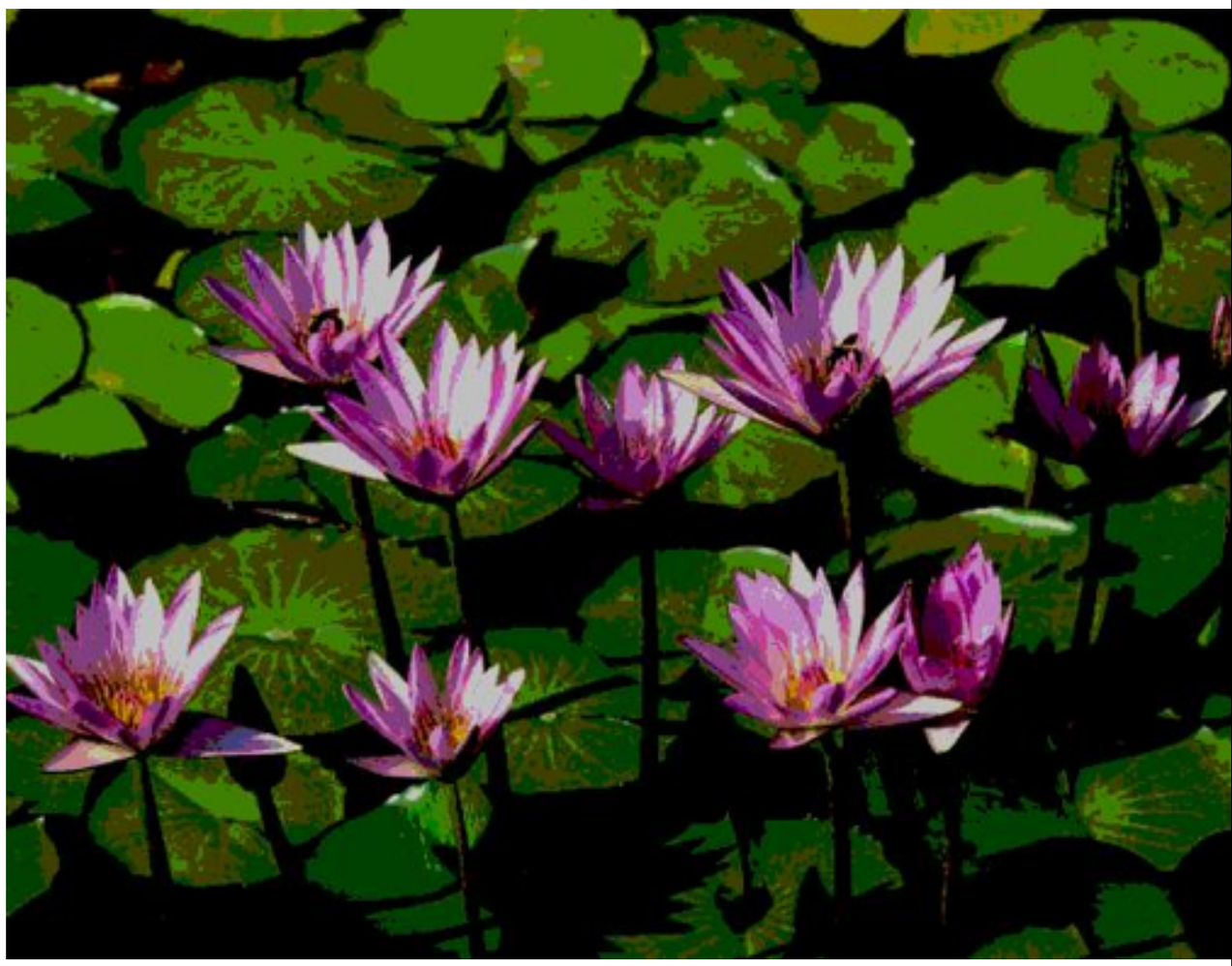
but there is no need to just perform linear convolution!

interesting operations

- the power of simple filtering
  - mathworks demo

but there is no need to just perform linear convolution!

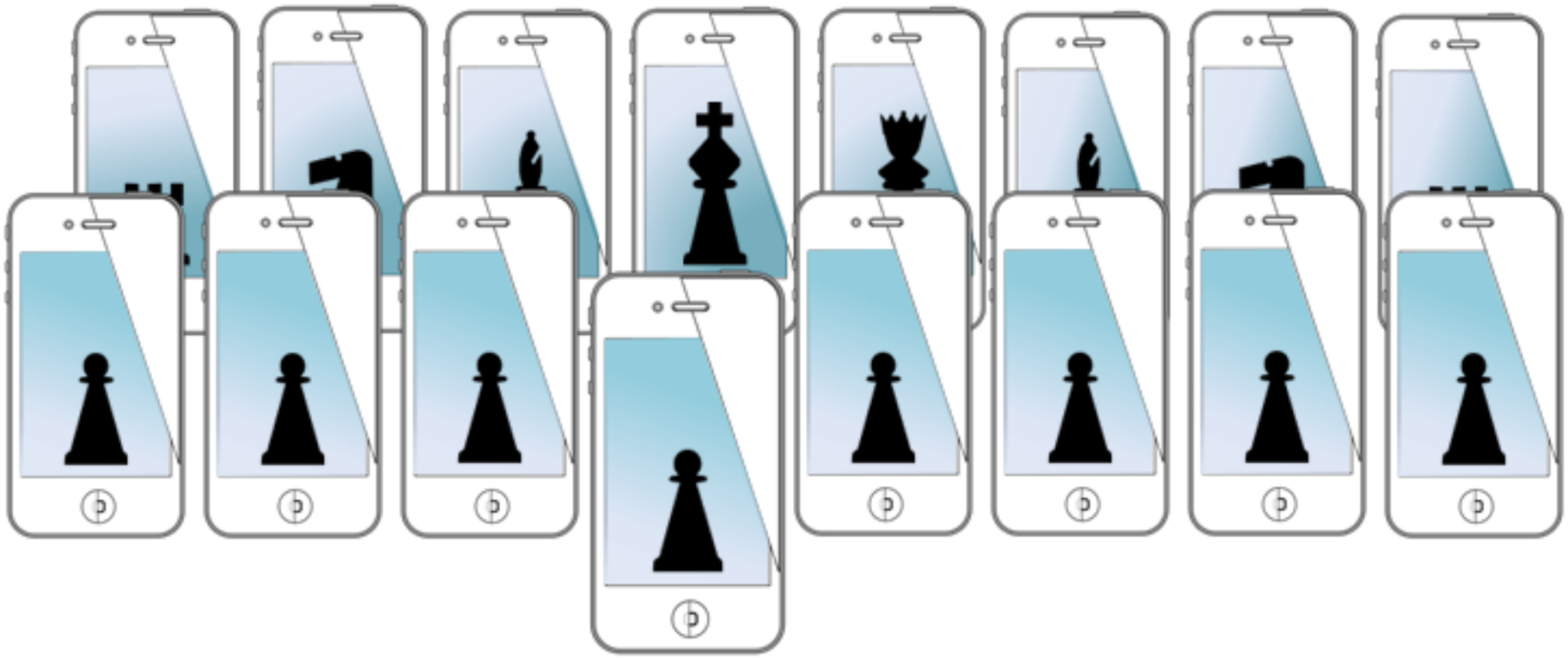




for next time…

- using core image
- video processing
  - detection and tracking of faces
- some computer vision

MOBILE SENSING LEARNING & CONTROL



CSE5323 & 7323

Mobile Sensing, Learning, and Control

lecture ten: core motion demo and image processing

Eric C. Larson, Lyle School of Engineering,
Computer Science and Engineering, Southern Methodist University