- Previous class:
  - Play with image (jpeg) files
  - 2-dimensional array—matrix
- Now:
  - Color images—3-dimensional array
  - Multi-media project

1

Grayness: a value in [0..255]

0 = black
255 = white

These are *integer* values
Type: `uint8`

150	149	152	153	152	155
151	150	153	154	153	156
153	151	155	156	155	158
154	153	156	157	156	159
156	154	158	159	158	161
157	156	159	160	159	162

2

Problem: produce a negative

3

Problem: produce a negative

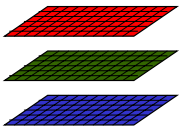
- "Negative" is what we say, but all color values are positive numbers!
- Think in terms of the extremes, 0 and 255. Then the "negative" just means the opposite side.
- So 0 is the opposite of 255;

1	…	254;
5	…	250;
30	…	225;
x	…	255-x

4

A color picture is made up of RGB matrices

Color image



3-d Array

`0 ≤ A(i,j,1) ≤ 255`
`0 ≤ A(i,j,2) ≤ 255`
`0 ≤ A(i,j,3) ≤ 255`

Operations on images amount to operations on matrices—good way to practice matrix manipulation!

7

Example: Mirror Image

`LawSchool.jpg`

`LawSchoolMirror.jpg`

8

Solution Framework

1. Read `LawSchool.jpg` from memory and convert it into an array.
2. Manipulate the Array.
3. Convert the array to a jpg file and write it to memory.

9

Reading and writing jpg files

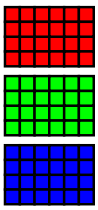
```
% Read jpg image and convert to
% a 3D array A
  A = imread('LawSchool.jpg');

% Write 3D array B to memory as
% a jpg image
  imwrite(B,'LawSchoolMirror.jpg')
```

10

A 3-d array as 3 matrices

```
[nr, nc, np] = size(A)  % dimensions of 3-d array A
```

#rows #columns #layers (pages)



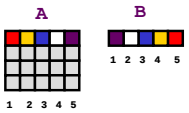
4-by-6 `M1= A(:,:,1)`

4-by-6 `M2= A(:,:,2)`

4-by-6 `M3= A(:,:,3)`

11

```
% Make B, a mirror image of A

[nr,nc,np]= size(A);
for r= 1:nr
  for c= 1:nc

      B(r,    )= A(r,         );

  end
end
```



12

```
% Make B, a mirror image of A

[nr,nc,np]= size(A);
for r= 1:nr
  for c= 1:nc

      B(r,c  )= A(r,nc-c+1  );

  end
end
```

13

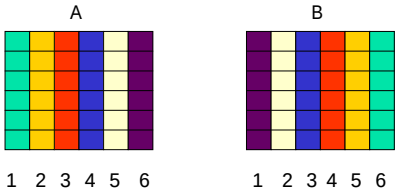
```
% Make B, a mirror image of A

[nr,nc,np]= size(A);
for r= 1:nr
  for c= 1:nc
    for p= 1:np
      B(r,c,p)= A(r,nc-c+1,p);
    end
  end
end
```

14

```
% Make mirror image of A -- the whole thing

A= imread('LawSchool.jpg');
[nr,nc,np]= size(A);

B= zeros(nr,nc,np);
B= uint8(B);  % Type for image color values

for r= 1:nr
  for c= 1:nc
    for p= 1:np
      B(r,c,p)= A(r,nc-c+1,p);
    end
  end
end
imwrite(B)  % Show 3-d array data as an image
imwrite(B,'LawSchoolMirror.jpg')
```

18

Vectorized code simplifies things…
Work with a whole column at a time



Column c in B
is column nc-c+1 in A

29

Consider a single matrix (just one layer)

```
[nr,nc,np] = size(A);
for c= 1:nc
    B( : ,c  ) = A( : ,nc+1-c  );


end
```

The colon says "all indices in this dimension." In this case it says "all rows."

31

Vectorized code to create a mirror image

```
A = imread('LawSchool.jpg')
[nr,nc,np] = size(A);
for c= 1:nc
    B(:,c,1) = A(:,nc+1-c,1)
    B(:,c,2) = A(:,nc+1-c,2)
    B(:,c,3) = A(:,nc+1-c,3)
end
imwrite(B,'LawSchoolMirror.jpg')
```

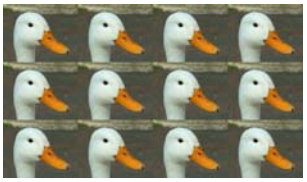
33

Turn the white duck yellow!

- The duck's body and the image's background show some contrast. However, neither the duck's body nor the background has a uniform color
- Are the RGB values different enough for us to write a "rule" in the program to tell between the duck and the background?
- Check out the RGB values!

41

Extracting subarrays and tiling



- Accessing a submatrix: M(_:_ , _:_)
- Accessing a subarray (3-d): P(_:_ , _:_ , :)
- Concatenate horizontally: [PL PR]
- Concatenate vertically: [PT; PB]

42