

# Designing An App ...

*Lawrence Snyder*  
*University of Washington, Seattle*

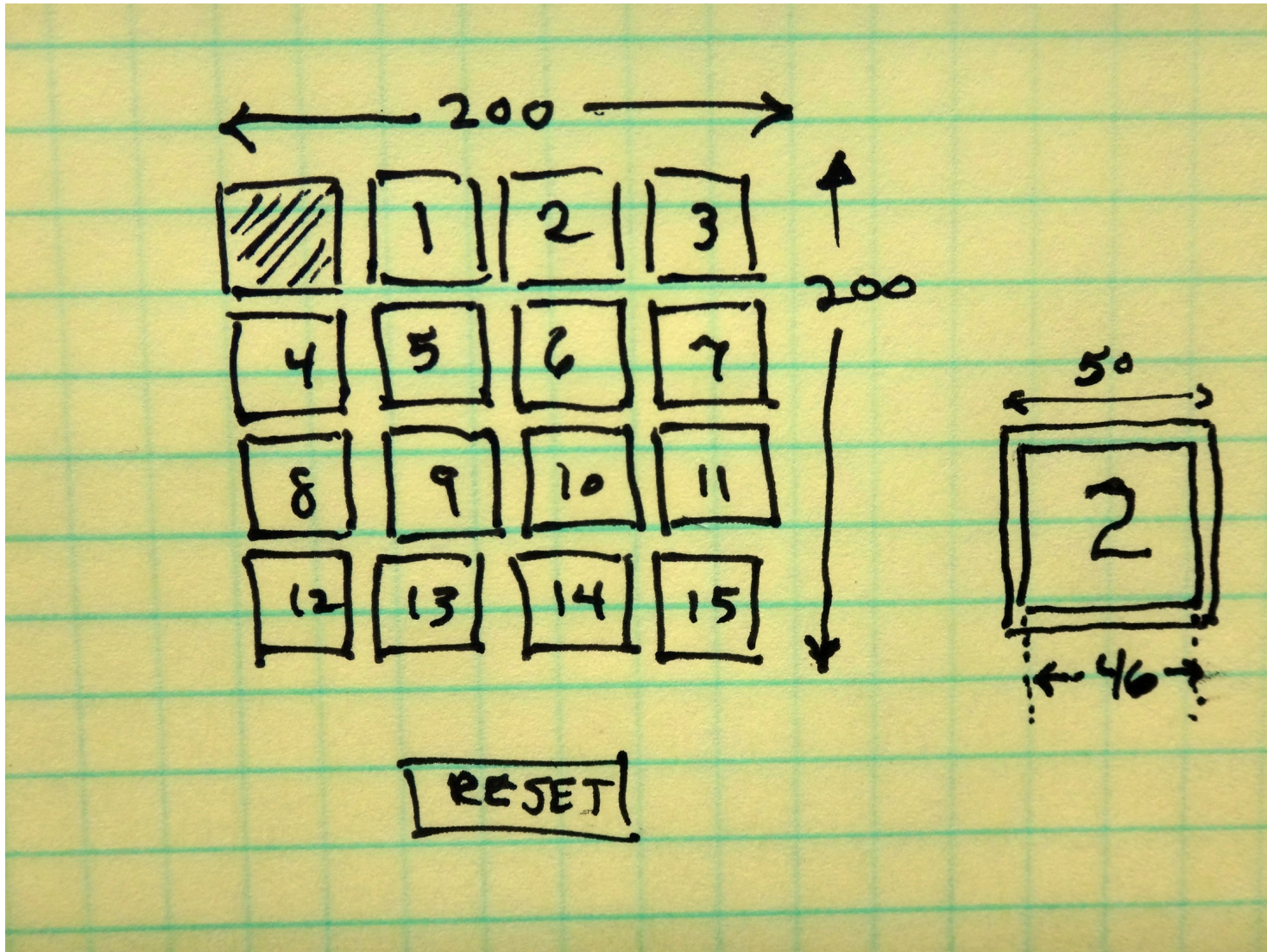
# Step-by-Step To Build A Puzzle

The app works like the children's toy. To move a piece, click on the one to be moved



	1	2	3
4	5	6	7
8	9	10	11
12	13	14	15

# Begin By Making Sketch ...

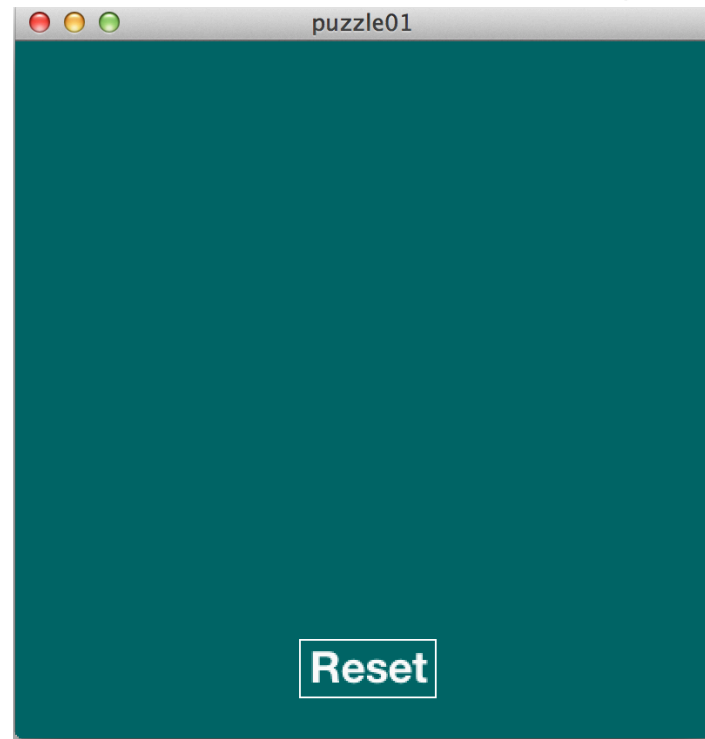


# Begin with Simplest Parts

```
PFont digits;
```

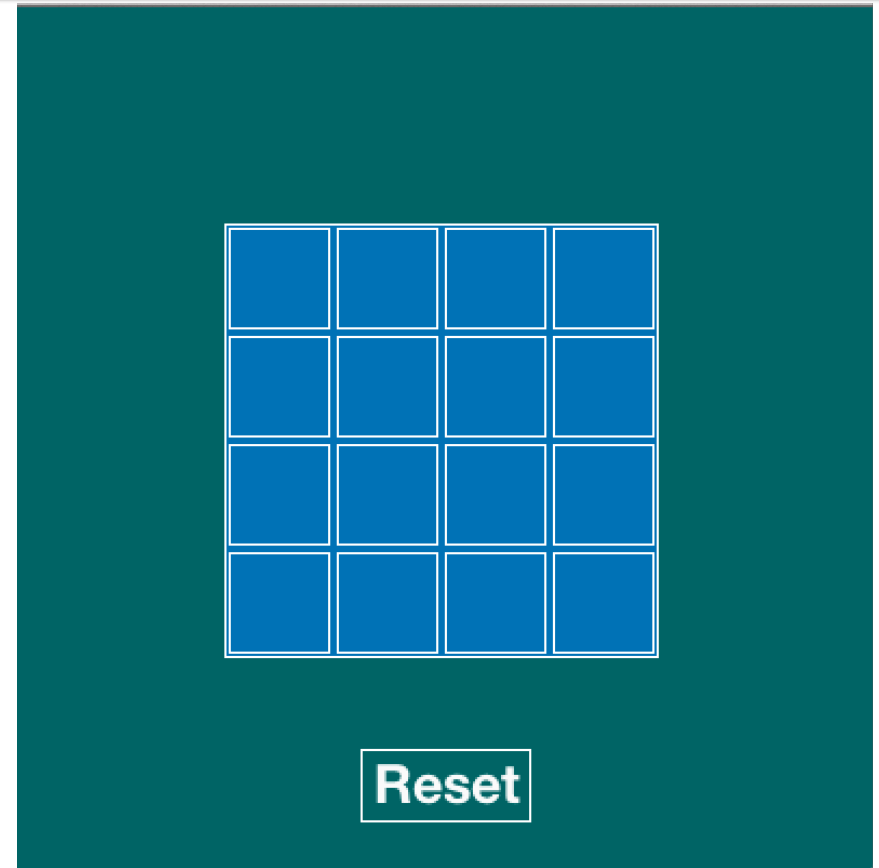
```
void setup( ) {  
  size(400,400);  
  background(0,100,100);  
  digits = loadFont("HelveticaNeue-Bold-25.vlw");  
  textFont(digits);  
}
```

```
void draw( ) {  
  fill(0,100,100);  
  stroke(255);  
  rect(163, 343, 78, 33);  
  fill(255);  
  text("Reset",169, 368);  
}
```



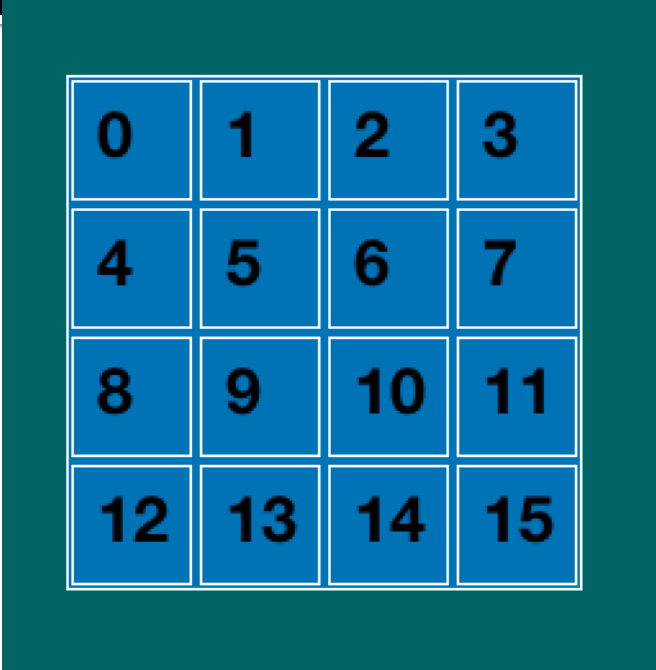
# Begin Drawing ... Use Functions

```
void draw( ) {  
    fill(0,100,100);  
    stroke(255);  
    rect(163, 343, 78, 33);  
    fill(255);  
    text("Reset",169, 368);  
    gameBd( );  
}  
  
void gameBd( ) {  
    fill(10,116,180);  
    rect(100, 100, 200,200);  
    for(int i=0; i<4; i++) {  
        for(int j=0; j<4; j++) {  
            rect(102+j*50, 102+i*50, 46, 46);  
        }  
    }  
}
```



# Fill In Numbers Into Squares

```
void gameBd( ) {  
    fill(10,116,180);  
    rect(100, 100, 200,200);  
    for(int i=0; i<4; i++) {  
        for(int j=0; j<4; j++) {  
            fill(10,116,180);  
            rect(102+j*50, 102+i*50, 46, 46);  
            fill(0);  
            text(str(i*4 + j), 102+j*50+10, 102+i*50+30);  
        }  
    }  
}
```



0	1	2	3
4	5	6	7
8	9	10	11
12	13	14	15

# Declare The Tile and Color Arrays

- The tiles[ ] array is initialized with digits, but it could have been assigned in a loop

```
PFont digits;  
int[ ] tiles = {0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15};  
color[ ] shade = new color[16];
```

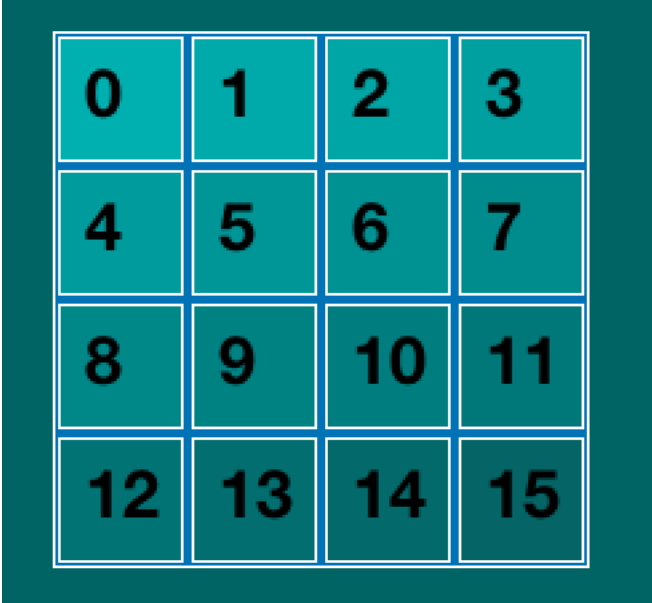
- The shade[ ] array is just a series of cells ... it is initialized in a loop in `setup()`

```
textFont(digits);  
for(int i=0; i<16; i++) {  
    shade[i] = color(15,175-5*i,175-5*i);  
}
```

# Returning To gameBd( ) ...

- We can now draw numbers from tiles[ ] with colors from shade[ ]

```
void gameBd( ) {  
    fill(10,116,180);  
    rect(100, 100, 200,200);  
    for(int i=0; i<4; i++) {  
        for(int j=0; j<4; j++) {  
            fill(shade[i*4 + j]);  
            rect(102+j*50, 102+i*50, 46, 46);  
            fill(0);  
            text(str(i*4 + j), 102+j*50+10, 102+i*50+30);  
        }  
    }  
}
```



0	1	2	3
4	5	6	7
8	9	10	11
12	13	14	15

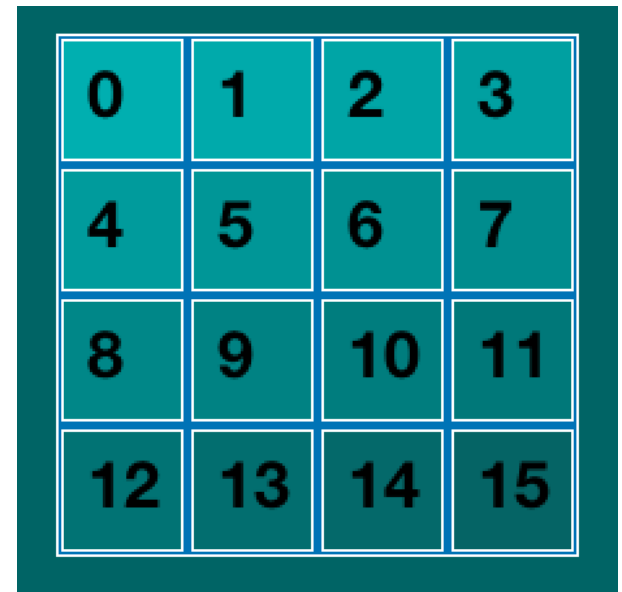


# Abstraction! Row, Col $\rightarrow$ Index

- convert from r/c ref.s to array index ref.s

```
int convert(int row, int col) {  
    return row*4 + col;  
}
```

```
for(int i=0; i<4; i++) {  
    for(int j=0; j<4; j++) {  
        fill(shade[convert(i,j)]);  
        rect(102+j*50, 102+i*50, 46, 46);  
        fill(0);  
        text(str(tiles[convert(i,j)]),  
             102+j*50+10, 102+i*50+30);  
    }  
}
```



0	1	2	3
4	5	6	7
8	9	10	11
12	13	14	15

# Avoid Displaying Zero

```
void gameBd( ) {  
    fill(10,116,180);  
    rect(100, 100, 200,200);  
    for(int i=0; i<4; i++) {  
        for(int j=0; j<4; j++) {  
            if(tiles[convert(i,j)] != 0) {  
                fill(shade[convert(i,j)]);  
                rect(102+j*50, 102+i*50, 46, 46);  
                fill(0);  
                text(str(tiles[convert(i,j)]),  
                    102+j*50+10, 102+i*50+30);  
            }  
        }  
    }  
}
```



	1	2	3
4	5	6	7
8	9	10	11
12	13	14	15

# Program The Action ...

- Like the birthday array, when we click on a tile, we compute the row and column as

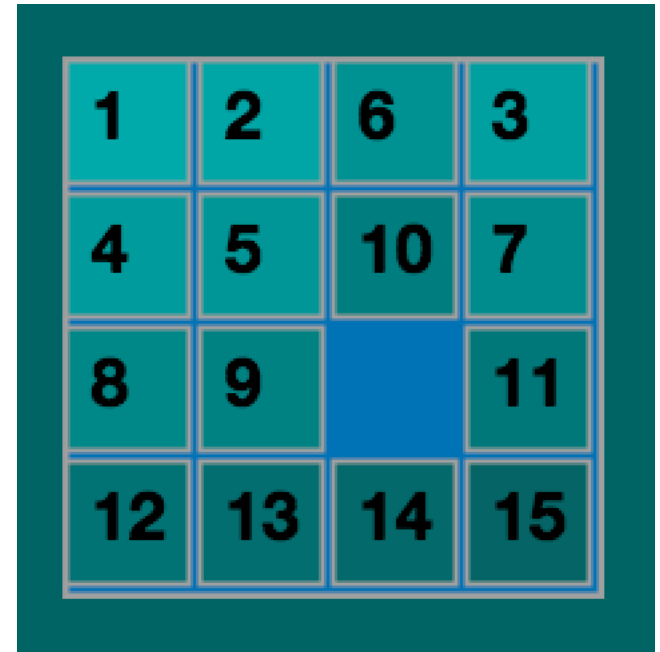
`row = (mouseY - 100)/50;`

`col = (mouseX - 100)/50;`

- If the present open position is at `ex,wy` then what are legal moves?

- The row or col can change by 1 but not both

- Say that as: `abs(row-ex) + abs(col-wy) == 1`



# Program The Action ...

- And if ( $\text{abs}(\text{row-ex}) + \text{abs}(\text{col-wy}) == 1$ ) then what?

- Exchange `tiles[ ]` values
- Exchange `shade[ ]` values
- ... for `ex,wy` with `row,col`

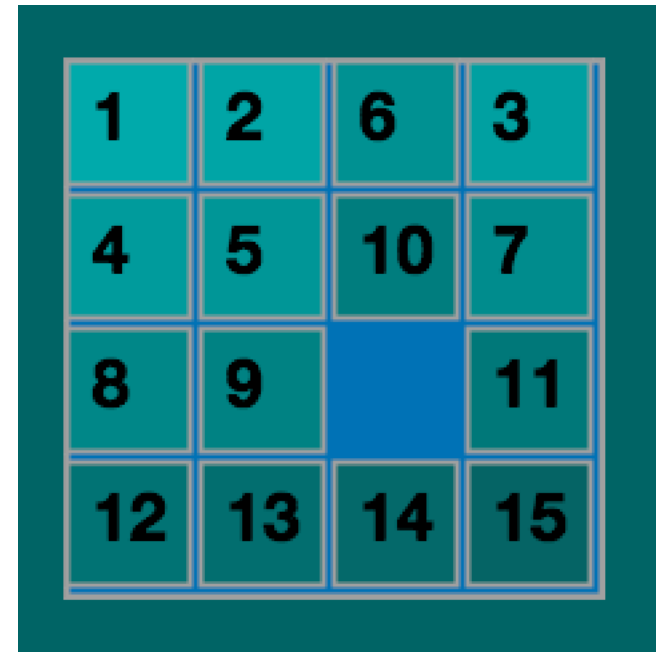
- An example exchange ...

- To exchange values of `a` and `b`

```
int temp = a;
```

```
a = b;
```

```
b = temp;
```

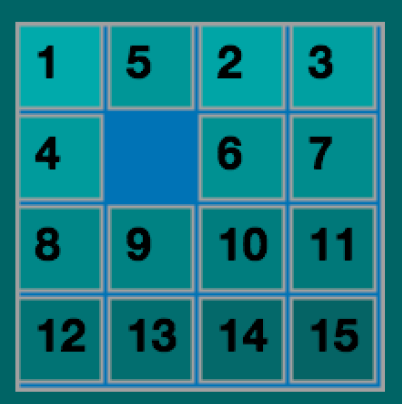


1	2	6	3
4	5	10	7
8	9		11
12	13	14	15

← Standard Technique

# New Function processes Clicks

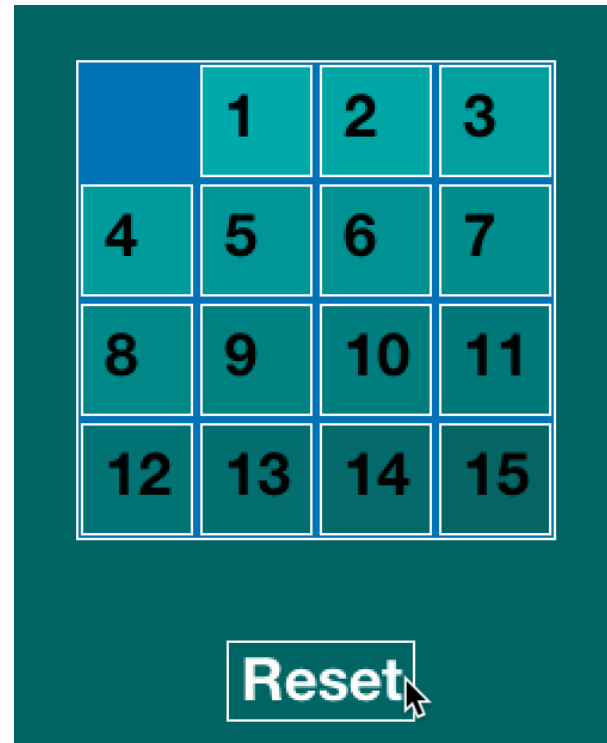
```
void mouseReleased( ) {  
    int row = (mouseY-100)/50;  
    int col = (mouseX-100)/50;  
    if ((abs(row-ex) + abs(col-wy)) == 1) {  
        //exchange tiles  
        int temp = tiles[convert(ex,wy)];  
        tiles[convert(ex,wy)] = tiles[convert(row,col)];  
        tiles[convert(row,col)] = temp;  
        //exchange colors  
        color tempc = shade[convert(ex,wy)];  
        shade[convert(ex,wy)] = shade[convert(row, col)];  
        shade[convert(row,col)] = tempc;  
        ex = row;  
        wy = col;  
    }  
}
```



1	5	2	3
4		6	7
8	9	10	11
12	13	14	15

# Reset ... Initialize tiles[], shade[]

```
void mousePressed( ) {  
    if(mouseX > 163 && mouseX < 163+78 && // Resetting?  
        mouseY > 343 && mouseY < 343+33) {  
        for(int i=0; i<16; i++) { // Yes, reinitialize  
            tiles[i] = i;  
            shade[i] = color(15,175-5*i,175-5*i);  
        }  
        ex = 0;  
        wy = 0;  
    }  
}
```



# Summary

- Sketched the idea on paper
- Started with simplest stuff
- Built on previous work by adding one function or one idea at a time
- Checked after every improvement