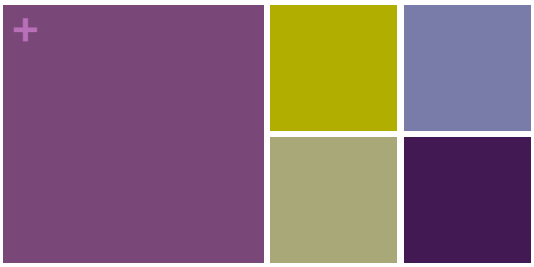


Scope, Tracing, and Basic Trigonometry

Review

- Parameterizing a shape
  - have a default size frame for your shape to fit in.
  - alter position relative to the reference point and scale
  - alter size relative to scale.
  - instead of scale
    - use a scaled reference size frame
- Functions and named constants improve readability, reusability, and scalability of your code.
- Variable Lifetime and Scope
  - global variables

Shadowing

- When there is a name conflict between variables of different scopes

```
int x = 10;
void setup() {
  int x = 5;
  int y = x;
}
```

- The conflicting variables can not have different types (or it's considered a re-declaration and is not allowed)
- When shadowed, smaller (inner) scopes have precedence over larger (outer) scopes

```
int v1 = 1;

void setup() {
  int v2 = 2;

  for (int v3=3; v3 <= 3; v3++) {
    int v4 = 4;
    println("------setup------");
    println(v1);
    println(v2);
    println(v3);
    println(v4);
    //println(v5);
  }

  int v3 = 6;
  println(v3);

  aFunction(v2);
}

void aFunction(int v5) {
  println("------aFunction------");
  println(v1);
  //println(v2);
  //println(v3);
  //println(v4);
  println(v5);
}
```

- What is printed?
- What happens if the second v3 declaration is removed?
- What would happen if the v5 print statement is executed?
- What would happen if commented statements in aFunction were called?

Example

- scopeLines

Code tracing

- We learn to read code by executing the code line by line
- Do not jump ahead
- Do exactly what the code says, step by step
- Keep a diagram of all variables and update them accordingly
- Mistakes are almost always due to skipping steps

Trace this

```
1  int n = 365;
2  int sum = 0;
3  int digit;

4  while(n>0) {
5    digit = n%10;
6    sum += digit;
7    n /= 10;
8  }

9  println(sum);
```

7

Nested loops

- You can put a loop within a loop
- Nesting levels are unlimited, but in practice programmers rarely go beyond 3
- Two loops nested is very common, especially when dealing with naturally 2-dimensional structures (grids)

-
```
for(...){
    for(...){
      }
  }
```
-
```
while(...){
    while(...){
      }
  }
```
-
```
for(...){
    while(...){
      }
  }
```
-
```
while(...){
    for(...){
      }
  }
```

Nested `for`

```
int i, j, end = 10;

for (i = 1; i <= end; i++) {
  for (j = i; j <= end; j++) {
    print("*");
  }
  println();
}
```

9

Nested `for`

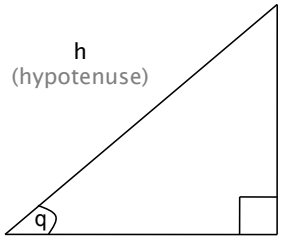
```
int i, j, end = 10;

for (i = 1; i <= end; i++) {
  for (j = 1; j <= i; j++) {
    print("*");
  }
  println();
}
```

10

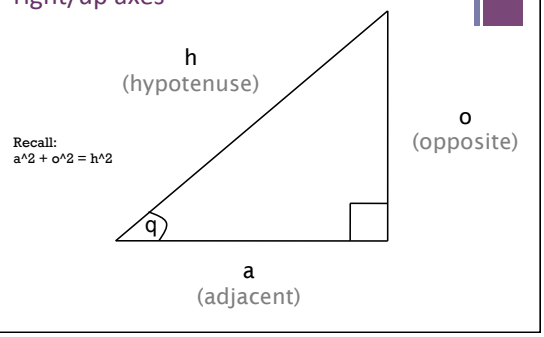
Examples

- indexTile (one loop)
- indexTile (loop with nested Loop)

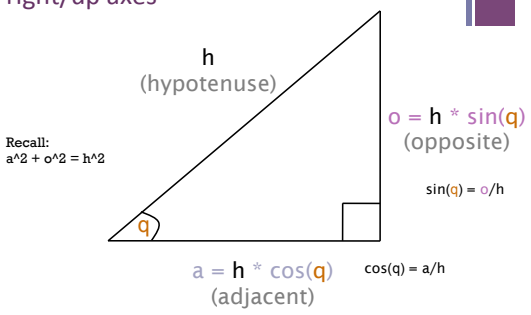
Basics of Trigonometry assuming right/up axes



Basics of Trigonometry assuming right/up axes



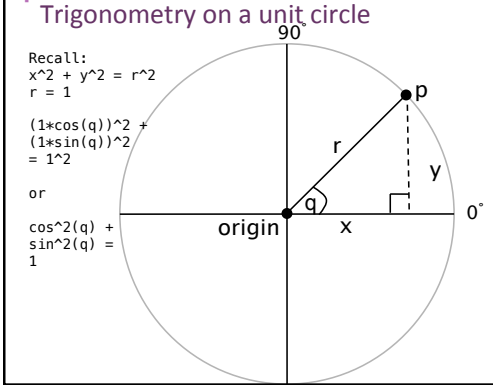
Basics of Trigonometry assuming right/up axes



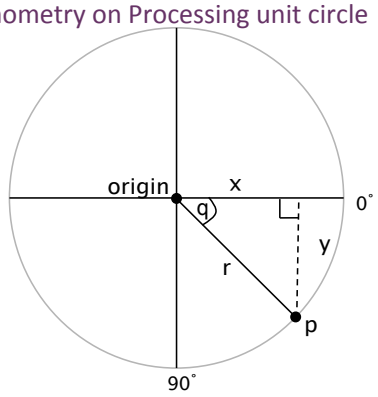
Definition

- sin(q) = o/h
- o = h*sin(q)
- cos(q) = a/h
- a = h*cos(q)
- tangent(q) = o/a = sin(q)/cos(q)

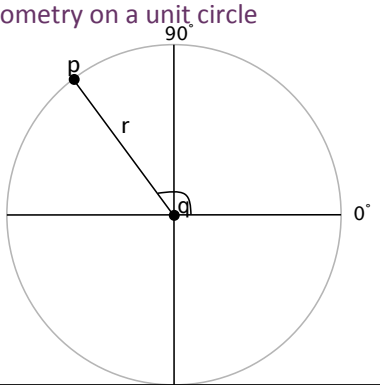
Trigonometry on a unit circle



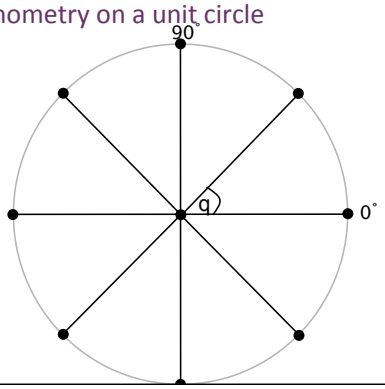
Trigonometry on Processing unit circle



Trigonometry on a unit circle



Trigonometry on a unit circle



Drawing points along a circle

```
int steps = 8;
int radius = 20;
float angle = 2*PI/steps;

for (int i=0; i<steps; i++) {
  float x = cos(angle*i)*radius;
  float y = sin(angle*i)*radius;

  // draw a point every 1/8th of a circle
  ellipse(x, y, 10, 10);
}
```

Examples

- points on a circle
- overlapping ellipses on a circle
- spokes
- polygon
- nested version (star)

Example: cyclical change

- Drawing a sine wave
- Using sine to manipulate height of an object
- Using cosine to manipulate width of an object