Tools for Aquarium and Word Clouds

Big Picture

- How do you go from specifications
- to code:

+ Big Picture

- How do you go from specifications
  - create an object that gives access to its position
- to code:

Big Picture

- How do you go from specifications
  - create an object that gives access to its position
- to code:
  - class TryOne {
    - float x,y;
    - public TryOne(float x, float y) {
    -   this.x = x;
    -   this.y = y;
    - }
    - public float getX() { return x;}
    - public float getY() (return y;}
  - }

Step 1: locate key phrases

- create an object that gives access to its position
- How do we create an object?
  - make a class
    - fields/attributes
    - constructor
    - methods

+ Step 1: locate key phrases

- create an object that gives access to its position
- How do we create an object?
  - make a class
    - fields/attributes
    - constructor
    - methods
- How do we give access?
  - accessor method to return an attribute

+ Step 1: locate key phrases

- create an object that gives access to its position
- How do we create an object?
  - make a class
    - fields/attributes
    - constructor
    - methods
- How do we give access?
  - accessor method to return an attribute
- How do we define position?
  - attributes that define location.

Step 2: Do each part

- create an object that gives access to its position
- make a class
  - class TryOne {
    - // what fields do we need?
    - TryOne() { // constructor
    -
    -
    - }
    - // what other methods do we need?
  - }

+ Step 2: Do each part

- create an object that gives access to its position
- make a class
  - class TryOne {
    - float x,y; // add attributes here
    - public TryOne(float x, float y) { // put attributes in constructor
    -   this.x = x;
    -   this.y = y;
    - }
    - // what methods do we need?
  - }

Step 2: Do each part

- create an object that gives access to its position
- make a class
  - class TryOne {
    - float x,y;
    - public TryOne(float x, float y) {
    - this.x = x;
    - this.y = y;
    - }
    - public float getX() { return x;} // give access with accessor
    - public float getY() (return y;} // give access with getter
  - }

Fitting your creature into specified space

- create an creature that gives access to its position and its size and can draw itself centered in its position and filling up a circle with diameter equal to its size

- 2 options, of many
  - option 1 use the size passed in and make all of your shapes to fit inside the specified size
  - option 2 make code for your object, then scale it and move it to fit in the expected size and location.

Option 2 (for AnimatedObject)

- We have a creature, but it's the wrong size.
  - we need to scale, however
    - we don't want the location to change
  - ideally, our creature, c, is drawn using position variables.
    - in that case the following algorithm should work
      - push matrix
        - translate to c.getX(), c.getY()
        - scale down relative to c.getSize()
        - draw creature at 0,0
      - pop matrix
    - test by drawing a bounding ellipse
      - with only a border with arguments
      - c.getX(),c.getY(), c.getSize(),c.getSize()
    - If the creature doesn't fit, then you can adjust your translation and scale as needed, but make sure you test with multiple sizes.

Specifics of algorithm

- how do we draw creature at 0,0
- if your code uses the creatures x and y position in each call for drawing:
  - ellipse(X + 0.15 * size, Y + 0.15 * size, .08 * size, .08 * size);
  - rect(X - 0.15 * size, Y + 0.15 * size, .08 * size, .08 * size);
- Option 1:
  - mask X and Y with local variables float X and float Y
  - float X = 0;
  - float Y = 0;
  - ellipse(X + 0.15 * size, Y + 0.15 * size, .08 * size, .08 * size);
  - rect(X - 0.15 * size, Y + 0.15 * size, .08 * size, .08 * size);

Specifics of algorithm

- how do we draw creature at 0,0
- if your code uses the creatures x and y position in each call for drawing:
  - ellipse(X + 0.15 * size, Y + 0.15 * size, .08 * size, .08 * size);
  - rect(X - 0.15 * size, Y + 0.15 * size, .08 * size, .08 * size);
- Option 2:
  - save X and Y with local variables float oldX and float oldY
  - float oldX = X;
  - float oldY = Y;
  - X = 0;
  - Y = 0;
  - ellipse(X + 0.15 * size, Y + 0.15 * size, .08 * size, .08 * size);
  - rect(X - 0.15 * size, Y + 0.15 * size, .08 * size, .08 * size);
  - … // finish creature drawing
  - X = oldX;
  - Y = oldY;

+
Example 1

- Drawing uses creature location, but not size:
  - pushMatrix();
  -    translate(x,y);
  -    scale(size/450.0);
  -    drawMagikarp(0, 0);
  - popMatrix();

Example 2 (use masking)

- Drawing uses creature location, but not size:
  - pushMatrix();
  - translate(x,y);
  - scale(size/300);
  - float x = 0;
  - float y = 0;
  - fill(0,0,155);
  - triangle(x, y, x+150, y+150, x+150, y-150);
  - triangle(x, y, x-150, y+150, x-150, y-150);
  - noStroke();
  - …
  - popMatrix();

Example 3 (use tempVar)

- Drawing uses creature location, but not size:

```
 pushMatrix();
   translate(x,y);
   scale(size/300);
    float oldX = x;
    float oldY = y;
    x = 0;
    y = 0;
  fill(0,0,155);
  triangle(x, y, x+150, y+150, x+150, y-150);
  triangle(x, y, x-150, y+150, x-150, y-150);
  noStroke();
…
popMatrix();
x = oldX;
y = oldY;
```

Example

- Let's look at our aquarium and fix one of the creatures.
  - The alien?

Signature

- make a signature to fit in a width and height assuming that 0,0 is the upper left hand corner.
- void signature(float w, float h)
- Need your name and the name of your creature.
- Need to adjust the font size based on width and the number of characters wide and high your string are.
  - Typically the width of a lowercase character is about half of the font size.
- text is drawn from the lower left hand corner as a reference point, not the upper left hand corner, so you need to adjust accordingly
  - text(0,h,"my signature");

Word Clouds exercise

- create a secondary filter so that your words have more meaning
- create a tiling of your choosing so that there is no overlap.

How do we approach this????

Word Clouds exercise

- create a secondary filter so that your words have more meaning
- create a tiling of your choosing so that there is no overlap.

locate key phrases

Secondary Filter

Let's look at our options:

- Stopwords
  - compare tokens with an array of stopwords, make a subset of tokens that has no stopwords.
- hastag removal
  - if(token[i].charAt(0) == '#') { // if it's a hashtag...
- topic words
  - only display words that are about a particular topic using a list or multiple lists of keepwords
- substring filter
  - remove or keep a word that contains a substring
  - if(token[i].contains("fun") { // if fun is in the word

+ Secondary Filter

Let's look at our options:

- Stopwords
  - compare tokens with an array of stopwords, make a subset of tokens that has no stopwords.
- hastag removal
  - if(token[i].charAt(0) == '#') { // if it's a hashtag...
- topic words
  - only display words that are about a particular topic using a list or multiple lists of keepwords
- substring filter
  - remove or keep a word that contains a substring
  - if(token[i].contains("fun") { // if fun is in the word

All of these require looping through the tokens

+

Secondary Filter

Let's look at our options:

- Stopwords
  - compare tokens with an array of stopwords, make a subset of tokens that has no stopwords.
- hastag removal
  - if(token[i].charAt(0) == '#') { // if it's a hashtag...
- topic words
  - only display words that are about a particular topic using a list or multiple lists of keepwords
- substring filter
  - remove or keep a word that contains a substring
  - if(token[i].contains("fun") { // if fun is in the word

All of these require looping through the tokens

Some also require looping through the filters

+ Other Filtering

locate key phrases

- Stopwords
  - compare tokens with an array of stopwords, make a subset of tokens that has no stopwords.
- hastag removal
  - if(token[i].charAt(0) == '#') { // if it's a hashtag...
- topic words
  - only display words that are about a particular topic using a list or multiple lists of keepwords
- substring filter
  - remove or keep a word that contains a substring
  - if(token[i].contains("fun") { // if fun is in the word

Stopwords Algorithm

```
have array of tokens
read array of stopwords
create array of filteredWords // subset of tokens
count = 0
for each token t
    boolean add = true
    for each stopword s
        if s.equals(t)
            add  = false
    if add // not a stopword
        filteredWords[count] = t;
        increment count
```

Other Filtering

locate key phrases

- Stopwords
  - compare tokens with an array of stopwords, make a subset of tokens that has no stopwords.
- hastag removal
  - if(token[i].charAt(0) == '#') { // if it's a hashtag…
- topic words
  - only display words that are about a particular topic using a list or multiple lists of keepwords
- substring filter
  - remove or keep a word that contains a substring
  - if(token[i].contains("fun") { // if fun is in the word

Hashtag Removal Algorithm

- create array of filteredWords
- count = 0
- for each token t
    - if(token[i].charAt(0) != '#')
        - filteredWords[count] = t;
        - increment count

Other Filtering

locate key phrases

- Stopwords
  - compare tokens with an array of stopwords, make a subset of tokens that has no stopwords.
- hastag removal
  - if(token[i].charAt(0) == '#') { // if it's a hashtag...
- topic words
  - only display words that are about a particular topic using a list or multiple lists of keepwords
- substring filter
  - remove or keep a word that contains a substring
  - if(token[i].contains("fun") { // if fun is in the word

+ Topic words keep Algorithm

```
read array of topic words
create array of filteredWords
count = 0
for each token t
    boolean add = false
    for each topic word s
        if s.equals(t)
            add  = true
    if add
        filteredWords[count] = t;
        increment count
```

+ Other Filtering

locate key phrases

- Stopwords
  - compare tokens with an array of stopwords, make a subset of tokens that has no stopwords.
- hastag removal
  - if(token[i].charAt(0) == '#') { // if it's a hashtag...
- topic words
  - only display words that are about a particular topic using a list or multiple lists of keepwords
- substring filter
  - remove or keep a word that contains a substring
  - if(token[i].contains("fun") { // if fun is in the word

Substring filter keep Algorithm

```
read array of substrings
create array of filteredWords
count = 0
for each token t
    boolean add = false
    for each substring s
        if t.contains(s)
            add  = true
    if add
        filteredWords[count] = t;
        increment count
```

+
Word Clouds exercise

- create a secondary filter so that your words have more meaning
- create a tiling of your choosing so that there is no overlap.

bullet 2

locate key phrases

Tiling with Random Arrangement

- While there are more tiles to place
  - get the next tile, t, to place
  - while(t is not placed)
    - set a random location, l, for the tile
    - if t does not intersect any previously placed tile
      - place t.

+ Tiling with Random Arrangement

- While there are more tiles to place
  - get the next tile, t, to place
  - while(t is not placed)
    - set a random location, l, for the tile
    - if t does not intersect any previously placed tile
      - place t.

Huh?

Tiling with Random Arrangement

- While there are more tiles to place
  - get the next tile, t, to place
  - while(t is not placed)
    - set a random location, l, for the tile
    - if t does not intersect any previously placed tile
      - place t.

locate key phrases

Tiling with Random Arrangement

- While there are more tiles to place
  - get the next tile, t, to place
  - while(t is not placed)
    - set a random location, l, for the tile
    - if t does not intersect any previously placed tile
      - place t.

We have a method for this.

locate key phrases

Tiling with Random Arrangement

- While there are more tiles to place
  - get the next tile, t, to place
  - while(t is not placed)
    - set a random location, l, for the tile
    - if t does not intersect any previously placed tile
      - place t.

What do we need here?

locate key phrases

Tiling with Random Arrangement

```
While there are more tiles to place
    get the next tile, t, to place
    while(t is not placed)
        set a random location, l, for the tile
        if t does not intersect any previously placed tile
            place t.
```

Maybe a loop?

locate key phrases

+

checking t against previously placed tiles

- basic idea
  - keep the index of the current item to place
  - randomly place the item at current index
  - loop from 0 to the current index and check if the place intersects
  - if not then increment current index (i.e. place the current item)

j

	0	1	2	3	4	5
	we	the	people	of	united	states
x	30	300	25			
y	30	35	25			
width	100	150	180	…		
height	100	50	30			

checking t against previously placed tiles

- basic idea
  - keep the index of the current item to place
  - randomly place the item at current index
  - loop from 0 to the current index and check if the place intersects
  - if not then increment current index (i.e. place the current item)

i		j			
0	1	2	3	4	5

	we	the	people	of	united	states
x	30	300	25			
y	30	35	25			
width	100	150	180	…		
height	100	50	30			

+

checking t against previously placed tiles

- basic idea
  - keep the index of the current item to place
  - randomly place the item at current index
  - loop from 0 to the current index and check if the place intersects
  - if not then increment current index (i.e. place the current item)

	i	j			
0	1	2	3	4	5

	we	the	people	of	united	states
x	30	300	25			
y	30	35	25			
width	100	150	180	…		
height	100	50	30			

checking t against previously placed tiles

- basic idea
  - keep the index of the current item to place
  - randomly place the item at current index
  - loop from 0 to the current index and check if the place intersects
  - if not then increment current index (i.e. place the current item)

j

	0	1	2	3	4	5
	we	**the**	**people**	**of**	**united**	**states**
x	30	300	30			
y	30	35	170			
width	100	150	180	…		
height	100	50	30			

+

checking t against previously placed tiles

- basic idea
  - keep the index of the current item to place
  - randomly place the item at current index
  - loop from 0 to the current index and check if the place intersects
  - if not then increment current index (i.e. place the current item)

	i		j			
	0	1	2	3	4	5
	we	the	people	of	united	states
x	30	300	30			
y	30	35	170			
width	100	150	180	…		
height	100	50	30			

checking t against previously placed tiles

- basic idea
  - keep the index of the current item to place
  - randomly place the item at current index
  - loop from 0 to the current index and check if the place intersects
  - if not then increment current index (i.e. place the current item)

	i	j			
0	1	2	3	4	5

	we	the	people	of	united	states
x	30	300	30			
y	30	35	170			
width	100	150	180	…		
height	100	50	30			

checking t against previously placed tiles

- basic idea
  - keep the index of the current item to place
  - randomly place the item at current index
  - loop from 0 to the current index and check if the place intersects
  - if not then increment current index (i.e. place the current item)

j

	0	1	2	3	4	5
	we	**the**	**people**	**of**	**united**	**states**
x	30	300	30			
y	30	35	170			
width	100	150	180	…		
height	100	50	30			

+

checking t against previously placed tiles

- basic idea
  - keep the index of the current item to place
  - randomly place the item at current index
  - loop from 0 to the current index and check if the place intersects
  - if not then increment current index
- details
  - for (int j = 0; j < sortedList.size(); j++)
    - while goodPlace == false
      - randomly place sortedList.get(j)
      - goodPlace = true
      - for(int i = 0; i < j; i++) {
        - if sortedList.get(i).intersects(sortedList.get(j))
          - goodPlace = false

+

Back to the exercise.

- …