Living Data Workshop 2

Link to slideshow: https://goo.gl/ytLXKB

https://goo.gl/ytLXKB

In this workshop...

Video

- Recap video requirements
- Today's activity: Feedback on your plan document

Processing

- Drawing with shapes
- sin/cos and radians

https://goo.gl/ytLXKB

Video Requirements

- 2 - 3 minutes in duration
  - If it is over 3 minutes, it will be marked up until 3 minutes
  - If your plan is for a longer video, ask for feedback where you can cut or consolidate information
- Can include *some* images/animations from other sources *along with* your video content, if the original creator is cited in the video.
- Citations should be included in the video when necessary
- Can be shot on your phone, but may include any technique you feel comfortable using, like animation, drawing and sketching.
- Check out this article for research about engaging education videos http://dl.acm.org/citation.cfm?doid=2556325.2566239

https://goo.gl/ytLXKB

Feedback Activity

https://goo.gl/ytLXKB

In this activity:

Organise yourselves into groups of 4

Each present your video plan in **3 minutes**, outlining:

- Your **topic** and content, what the video is about,
- Your intended **audience**, who the video is made for,
- The **style** you chose, how you are presenting to the audience
- The way your video fits the **narrative structure**

Spend 3 minutes discussing feedback.

Upload your feedback to Canvas (either now, or tonight)

https://goo.gl/ytLXKB

Uploading to Canvas

Due at *Midnight on Sunday*

- Expectation is that you do them in class
- OR you can write notes and upload later
- Extended date in case wi-fi or Canvas fails over during class time

Fill out feedback in the Canvas quiz titled *Draft video plan: peer review*

Fill out the questions for each peer, if there's empty questions at the end, Canvas will warn you there are unanswered questions, but that's ok.

https://goo.gl/ytLXKB

Giving Feedback

- **Be polite**: this is not a blind review, your peer will see your name with your feedback.
- **Be clear**: don’t write an essay, suggest WHERE and WHAT may be improved and HOW.
- **Be fair**: the video is not a reflection of your peer as a person, so your feedback should be about the video, not the person presenting.

Taking Feedback

- **Be receptive**: feedback is an opportunity, not a threat.
- **Be empathetic**: if your viewer doesn’t understand something you’re explaining in-person, will your audience understand it in a video?
- **Be decisive**: this feedback session gives you helpful suggestions, not mandatory instructions.

https://goo.gl/ytLXKB

Let’s get going!

1. 3 minute Peer Presentation
2. 3 minute discussion, upload comments
   https://goo.gl/5WeM4Z - animated flashcards/question prompts
3. Next presenter!

After This week’s class:

4. Consider feedback
5. Make changes
6. Shoot video
7. Bring draft video to next workshop (in 2 weeks)

https://goo.gl/ytLXKB

Processing

https://goo.gl/ytLXKB

Catch-up videos

- Loops https://youtu.be/qgGyKT0W7qM
- map() https://youtu.be/ynTvv025-Mk
- Table https://youtu.be/VcjC3WbpSgk

https://goo.gl/ytLXKB

What we are covering today

- ```
  beginShape();
  vertex();
  endShape();
  ```

- ```
  sin();
  cos();
  radians();
  ```

- Add it all together to show some data

https://goo.gl/ytLXKB

Drawing Shapes

`beginShape();` tells Processing to get ready to draw a new shape.

`vertex(x, y);` describes where a point on the shape is.

`endShape();` finishes the shape.

`endShape(CLOSE);` fishes the shape AND draws the line between the first and last vertices.

Example

```
beginShape();
vertex(50, 10);
vertex(75, 80);
vertex(20, 30);
vertex(80, 30);
vertex(25, 80);
endShape(CLOSE);
```

https://goo.gl/ytLXKB

`sin()`, `cos()` and `radians()`

`sin(x);` and `cos(y);`

- return a value betwen 0 and 1, which is the sin/cos of the angle you give it
- Are used to draw a circle without the circle function
- The value you give must be converted to radians

https://goo.gl/ytLXKB

Circle without the circle function

```
beginShape();
for (int i = 0; i < 360; i++) {
  float xCoord = width/2 + sin(radians(i)) * 40;
  float yCoord = height/2 + cos(radians(i)) * 40;
  vertex(xCoord, yCoord);
}
endShape(CLOSE);
```

https://goo.gl/ytLXKB

Circle without the circle function

```
beginShape();
for (int i = 0; i < 360; i++) {
  float xCoord = width/2 + sin(radians(i)) * 40;
  float yCoord = height/2 + cos(radians(i)) * 40;
  vertex(xCoord, yCoord);
}
endShape(CLOSE);
```

Diameter of circle
Center of circle
How far around to go each step.

Hint:
Change

```
i++
```

to

```
i+= 60
```

To make a different shape

https://goo.gl/ytLXKB

Radians

We need to convert the value we give `sin()` or `cos()` from degrees to radians.

There are two ways to convert our angle to radians

https://goo.gl/ytLXKB

These two have the same result

```
// angle to convert
float x = 30;
float xRadians = radians(x);
println(xRadians);
```

```
// angle to convert
float x = 30;
// map from 360 degrees to TWO_PI (radians)
float xRadians = map(x, 0, 360, 0, TWO_PI);
println(xRadians);
```

0.5235988 appears in the console in both cases

https://goo.gl/ytLXKB

These two have the same result

```
// angle to convert
float x = 30;
float xRadians = radians(x);
println(xRadians);
```

```
// angle to convert
float x = 30;
// map from 360 degrees to TWO_PI (radians)
float xRadians = map(x, 0, 360, 0, TWO_PI);
println(xRadians);
```

0.5235988 appears in the console in both cases

https://goo.gl/ytLXKB

So why use map?

Let's say we are drawing inside a loop, on a scale from 0 to numRows

```
for (int i = 0; i < numRows; i++) {
  // map i from 360 degrees to TWO_PI (radians)
  float angle = map(i, 0, numRows, 0, TWO_PI);
  println(angle);
}
```

This would be useful to display any of the nutrition data, the maximum amount of activity you got, or the average steps

https://goo.gl/ytLXKB

So why use `map()`?

Let's say we have some data, on a scale from 0 to 1

```
// data to convert
float inputData = 0.083;
// map from 360 degrees to TWO_PI (radians)
float mappedData = map(inputData, 0, 1, 0, TWO_PI);
println(mappedData);
```

A lot of the activity data from your tracker is from 0 to 1:
This includes *proportion* of time you were sedentary, or when you were doing light, moderate or vigorous activity.

https://goo.gl/ytLXKB

So why use map?

Let's say we have some data, on a scale from 0 to maxSteps

```
// data to convert
float inputData = 0.083;
// map from 360 degrees to TWO_PI (radians)
float mappedData = map(inputData, 0, maxSteps, 0, TWO_PI);
println(mappedData);
```

This would be useful to display any of the nutrition data, the maximum amount of activity you got, or the average steps

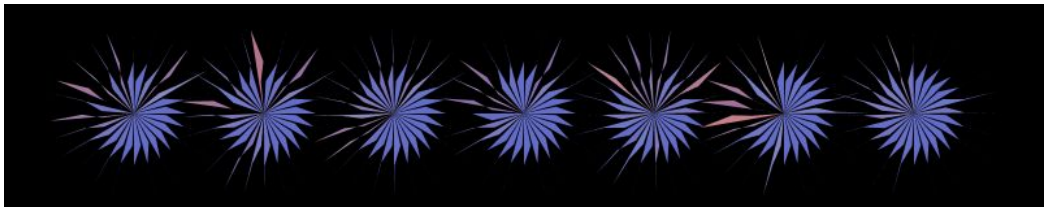
https://goo.gl/ytLXKB

Looking in context

Example sketches

https://goo.gl/gp9WBL

Phil's activity:



Emma's activity