DATA SCAPE WORKSHOP
ART 4990 AMFD

F_24

START: 08.27.24

END: 08.27.24

COURSE WWW

DATA SCAPING

PROJECT DESCRIPTION

In this project, we will each observe, collect, and organize data via Google Trends, and translate the selected data points into a 3D self-representative data informed landscape.

This landscape can be 3D printed to produce a physical object from your code.

You will need to install Processing to complete this project.

Follow along with the DATA_SCAPE_TEMPLATE demo in class. The demo folder contains notated code that we will walk through together to explain functionality, use, and modification.

The raw text code for the primary demonstration can also be found in this document.

DELIVERABLES

- **PNG IMAGE OUTPUT**
- **OBJ FILE OUTPUT**

TIMELINE

08.27.24-08.27.24

DATA SCAPE WORKSHOP

ART 4990 AMFD

F_24

START: 08.27.24

END: 08.27.24

COURSE WWW

```
////WELCOME TO DATA SCAPES!
////This template is for visualizing .csv data tables into landscape formats.
////Code by Aubrey Pohl, foundation of shape structure from Daniel Shiffman
////make sure all data and images to be used are stored in a data folder within the
sketch folder
////Leave all lines with "////" these are notes / comments
////Lines with "//" can be toggled on and off by removing the slashes or adding them


////import library to render data scape to a .obj file (3D printable ;)
import nervoussystem.obj.*;
boolean record = false;

////Variable for data table
Table table;

////Variable for image, to be used as a texture in this case.
PImage img, img2;

////Change this value to manipulate where in the .csv you are grabbing data to manipu-
late the image for texture
int imgManip = 0;

////Global variables for x and y plotted points of datascape form
int cols, rows;

////This value adjusts the scale of the landscape form. Adjust accordingly.
float scl = 25;

////caps determines the height of the peak structure of the data points.
////change this number accordingly to fit scape within frame
////similar to changing scale (scl)
    float caps = 25;

////Array sets to map and plot the points (data) to arrays for the x and y axis
float[][] terrain;

float mx, my;

void setup() {
  size(800, 800, P3D);
  //fullScreen(P3D);
  background(0);

////copy name of .csv table into first set of quotations below
  table = loadTable("", "header");

////image name and file extension
////turn on loadImage to use an image in the data folder.
////turn off loadImage if you want to create an image from scratch
  //img = loadImage("EARTH.png");

////grabbing the length of the .csv table / number of rows
  int rc = table.getRowCount();

////telling the global column and row variables to reflect the rowcount of data table
  cols = rc;
  rows = rc;

////telling the data scape form to be the size of the rowcount (this is why scale has to
be changed from time to time);
  terrain = new float[cols][rows];

////toggle this line to check the length of your csv rowcount
  println(rc);

////telling P3D the applied texture will be an image, and how to wrap it
  textureMode(IMAGE);
  textureWrap(CLAMP);
}


void draw() {
  background(0);

  mx = width/2;
  my = height/2;

  fill(150);
  noStroke();
```

DATA SCAPE WORKSHOP

ART 4990 AMFD

F_24

START: 08.27.24

END: 08.27.24

COURSE WWW

```
  //stroke();
  //strokeWeight();

////start of loops for x an y data plotting
////Here we start to plot the x and y values of the data and form
////X and Y are flipped in this case because it is easier to visualize this way in P3D
  for (int y = 0; y < rows; y++){
  for(int x = 0; x < cols; x++){

////this is where we grab the data points from .csv;
////copy the column name you want to grab points from into the quotations
////you can use the same column for x and y for symmetry, or different for asymmetry.

////use these for rough edges, whole numbers only
    //int num = table.getInt(x, "");
    //int num2 = table.getInt(y, "");

////use these for smoother data mapping, floating numbers (decimals)
    float num = table.getFloat(x, "");
    float num2 = table.getFloat(y, "");

////these variables plot the data points we just grabbed to the x and y axis
    float m = map(y, 0, cols, 0, num);
    float m2 = map(x, 0, rows, 0, num2);

////variables for manipulating the datascape with some animation. Optional use.
    //float manip = sin(frameCount*.0005*m);
    //float manip2 = sin(frameCount*.0005*m2);


////here we fill the arrays for x and y with our data points
    terrain[x][y] = map((m)+(m2), 0, 1, 0, caps);
    //terrain[x][y] = map((num)+(num2), 0, 1, 0, caps);
    //terrain[x][y] = map((m+manip)+(m2+manip2), 0, 1, 0, caps);

  }
  }
////end of loops for x and y data plotting

////Here we translate the 0 point and rotation of the sketch to make it more visible
////I do not recomend changing these value, but feel free to do so at your own risk
  translate(width/2, height);
  //rotateY(PI*frameCount*.005);
  rotateY(PI/1.335);
  rotateX(PI/2);

////here we create lights in order to show better the 3D form of the scape
////activates light render
  lights();
////directionalLight(Rvalue, Gvalue, Bvalue, x axis, y axis, z axis);
  //directionalLight(255, 255, 255, width/2, 0, -width/2);
  //directionalLight(255, 255, 255, 0, height, 0);
  directionalLight(255, 255, 255, width, 0, 0);

////toggle this on to export data scape to 3D .obj file
  if (record) {
    beginRecord("nervoussystem.obj.OBJExport", "DATA_SCAPE.obj");
  }


////This section is for creating an image from scratch, or manipulating an image with
data as texture
  //int num = table.getInt(imgManip, "");

  //img2 = createImage(width, height, RGB);
  //  img2.loadPixels();

  //  float lc = img2.pixels.length;

  //  for(int i = 0; i < lc; i++){


  //  img2.pixels[i] = i*num;
  //  }

  //  img2.updatePixels();


////start of loop for y of shape
  for(int y = 0; y < rows-1; y++){
```

DATA SCAPE WORKSHOP

ART 4990 AMFD

F_24

START: 08.27.24

END: 08.27.24

COURSE WWW

```
////the shape begins and ends outside of the x loop
////leave it this way, it causes the form to render smoother and properly
    beginShape(TRIANGLE_STRIP);

////toggle this line to add an image as texture
    //texture(img);
    //texture(img2);

////start of loop for x of the shape
    for(int x = 0; x < cols-1; x++){

////the u and v values are necessary for mapping an image as texture;
////they can change depending on the shape of the image
////manipulate at will for unexptected results
////turn off u v to export 3D .obj
    float u = (x);
    float v = (y);

////viola, we are creating the data scape by looping and mapping these vertices
////here you see where scl (scale) is used to manipulate the form
////I recommend leaving this seciont as is as to not lose the form completley
    //vertex(x*scl, y*scl, terrain[x][y], u, v);
    //vertex(x*scl, (y+1)*scl, terrain[x][y+1], u, v);

////to export 3D .obj, turn off above vertices and turn these on
    vertex(x*scl, y*scl, terrain[x][y]);
    vertex(x*scl, (y+1)*scl, terrain[x][y+1]);

    }
////end of x loop

////creating shape
    endShape(CLOSE);

  }
////end of y loop

////toggle this on/of to export 3D .obj file
  if (record) {
    endRecord();
    record = false;
  }
}


//// this function activates when the mouse is moved, and is allowing us to use the cam-
era function.
void mouseMoved(){

//// here we are changing a variable we have previously declared. This lets us use dif-
ferent manipulators at different points in the same sketch.
  mx = mouseX;
  my = mouseY;

//// the camera funciton lets us have a orbit control view of our data scape.
//// toggle on or off to change this, or change values to get a different view.
//// camera(eyeX, eyeY, eyeZ, centerX, centerY, centerZ, upX, upY, upZ);
  camera(mx, my, (height/2) / tan(PI/6), width/2, height/2, 0, 0, 1, 0);

}

void keyPressed(){

  if(key == ' '){
  String date = new java.text.SimpleDateFormat("yyyy_MM_dd_kkmmssSSS").format(new java.
util.Date());
    save("output/DATA_SCAPE_"+"_"+date+"_"+".png");
  }

////toggle on/off to export 3D .obj file for 3D printing
  if(key == 'r'){
    record = true;
  }


}
```