Code Specification

ArduinoAllen and ArduinoGates are done in **Arduino IDE**. Constellation.pde and Circle.pde are done in **processing**. All of the files are **coded in Java**.

ArduinoAllen

int fsrAnalogPin = 0;
int fsrReading; // the analog reading from the FSR resistor

// The method is called once at the very beginning that sets up the serial number of the Arduino and tries to establish connection
// with the processing
void setup()

// This method is automatically initiated and continuously keeps reading in the fsrAnalogPin by using the analogRead().
// Once the reading is between certain range (still need to be tested on), the Arduino should send the data to the processing
void loop()

ArduinoGates

int fsrAnalogPin = 0;
int fsrReading; // the analog reading from the FSR resistor

// The method is called once at the very beginning that sets up the serial number of the Arduino and tries to establish connection
// with the processing
void setup()

// This method is automatically initiated and continuously keeps reading in the fsrAnalogPin by using the analogRead(). Once the reading is between certain range (still need to be
// tested on), the Arduino should send the data to the processing
void loop()

Serial Communication

Serial Communication

Constellation (.pde)

/**The Constellation class takes the sensor data from two Arduinos as inputs and utilize the public Circle class to output the real-time data visualization animation. We utilize the dynamic constellation as the model to visualize the data where the dots represent people who already voted by triggering the sensor and the link between dots represent the connection between people, which will help create a sense of belongingness.
*/
import processing.serial.*;
import processing.sound.*;

ArrayList<Circle> circles; // a list of circles to be displayed
SoundFile file; // the sound effect that will be evoked
PFont prompt, current; // the text will be displayed
int count; // the # of people who have voted
Serial AllenArduino, GatesArduino // two Arduinos
String data1, data2; // data that Arduino send

// this method is called once at the very beginning that
// sets up the canvas where we can display the animation and
// setup the serial that we can receive data from the two Arduino
void setup()

// this method is automatically initiated and draw all the real-time data visualization:
// dots will be pulled by the gravitational force so that will be flying around in the screen.
void draw()

// Adds a dot on the animation once someone triggers the sensor in Allen building
void serialEvent(Serial AllenArduino)

// Adds a dot on the animation once someone triggers the sensor in Gates building
void serialEvent(Serial GatesArduino)

public class used in constellation

Circle.pde

PVector location // location of the dot
PVector velocity // velocity of the dot
PVector acceleration // acceleration of the dot
float G // gravitational constant of the universe
float mass // mass of the dot
float size // size of the dot

// Constructor that specifies the initial location, velocity,
// acceleration, mass and size of the dot and set the gravity
Circle (String building)

// Applys the given force to the dot to change its
// corresponding acceleration
void applyForce(PVector force)

// Applies the acceleration to the velocity of the dot thus changing its current location.
void update()

// displays the dot by setting its color, location, and size.
// Has its size shrink to 8
void display()

// displays the link between dot by setting the link's color,
// stroke weight. Displays the link only when the distance
// between two dots are between 250 and 350 pixels.
void link()

// calculates and returns the force of the gravity of the dot
// based on the gravitational force formula: G * mass * mass / (distance * distance)
PVector attract(Circle c)

// Check whether the dot approaches the edges of the
// scree. If it does, reverse its direction to make sure it always stays in the screen.
void checkEdge()