| Laelaps and the Teumessian Fox |

// A collaborative Project made by Emily Beckius, Lawrence Chen, Adrian Esqueda, Yichen Li, and Wen Ying Liu.

· concept ·

Traditionally, myths are verbally or textually documented and communicated. In the age of hyper-media storytelling enabled by the World Wide Web, linear, verbal means of storytelling is becoming obsolete. The concept of our project is to recreate and recount this ancient Greek myth that is suitable to modern audience by using multimedia and modern technology, such as programming, robotics, and augmented reality. We wanted to demonstrate the capabilities of augmented reality while bringing what was once considered fantasy into a plane of existence in which we can observe the myth here and now, corporealizing the ethereal and immortalizing the ephemeral.

· implementation :

Creating the markers:

Our project was split into several parts. For Part I, we planned on an augmented environment using AR.js, which we could implement using html. We tried creating custom markers through one of Jerome Etienne's libraries, but it didn't work. The library he created was outdated and flawed. So instead of creating custom markers, we resorted to two preset markers that our AR code could recognize.

Creating the augmented environment:

All five of us made a tree using Unity or Rhino. After creating the trees, we had to convert it into OBJ files. Once we did Adrian combined the trees that looked the best and grouped them together, making it seem like a forest.

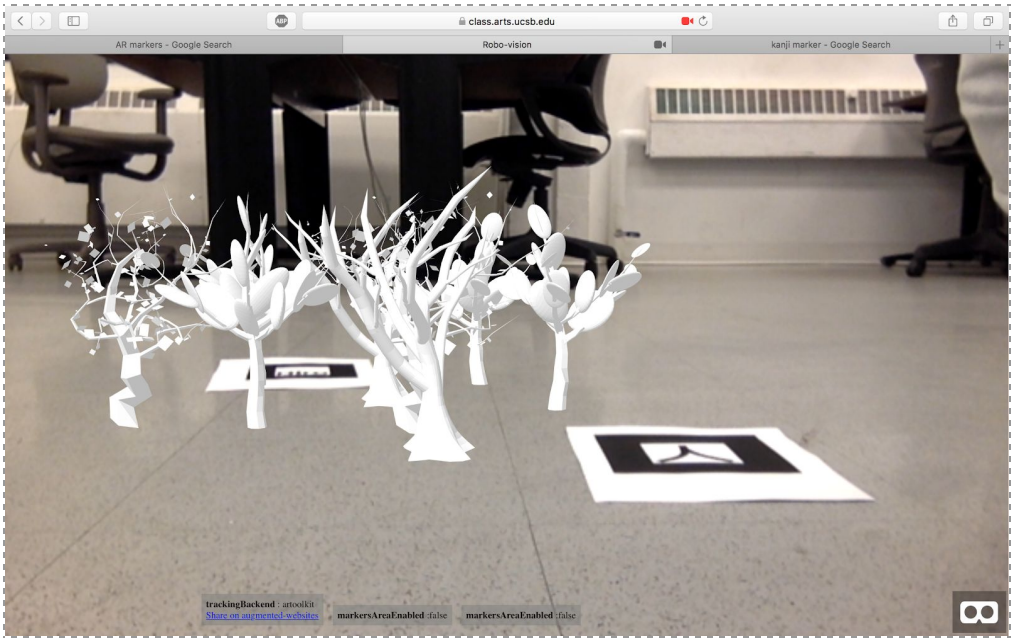
In the AR code (shown on page 9), we programmed the our webcam to recognize the two preset markers, so when it does, the augmented forests would appear on screen.

Camera:

For Part II, we planned on using the Raspberry Pi's webcam to livestream, scan, and recognize the markers we laid out, so that when we connect the pis' cameras to the class projector, we could see an augmented environment on the giant screen. To do that, we had to connect to the Raspberry Pi's camera.

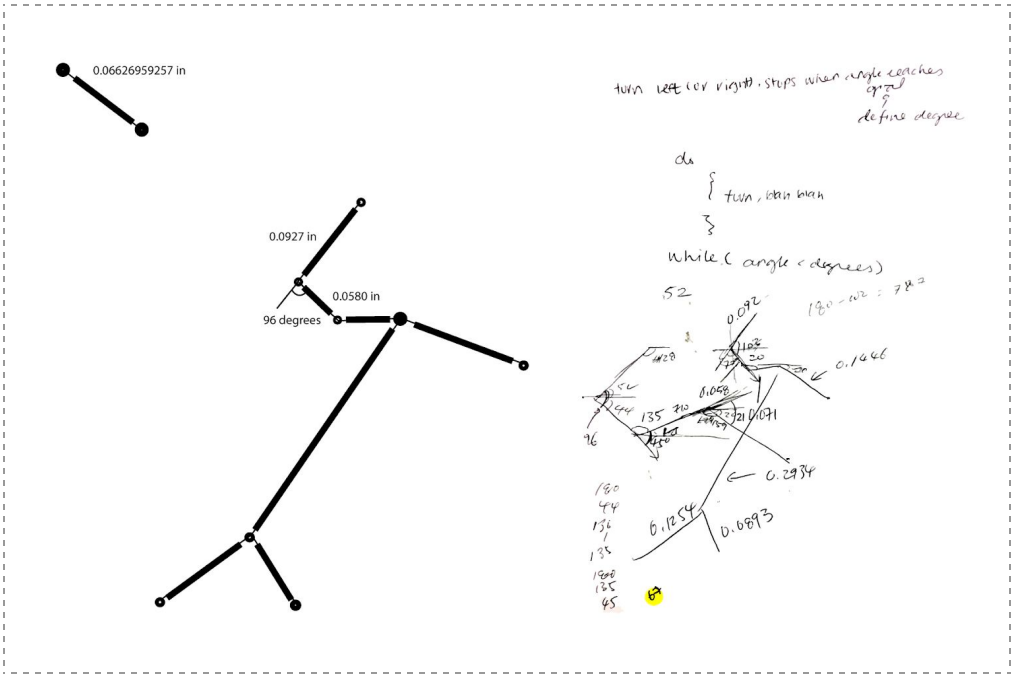
Wen spent several weeks trying to figure out how to connect to the Raspberry Pi's webcam. First, she connected to the pi's browser, and opened up the html file that had the ar.js code onto pi's web browser. While she was able to get into the Raspberry Pi's web browser, and while she was able to execute js files on the pi's terminal, she was unable to access the pi's webcam through the pi's browser. The pi's browser didn't work the same way her laptop's does.

Luckily, Rodger created a python file that allowed Wen to access the pi's webcam. When she got the code, all she had to do was connect to Robotswarm's wifi, log into the pi's browser, and execute the python code through terminal. She was able to connect to the pi's webcam after that, and we were able to livestream through its camera. Unfortunately, because of the webcam's framerate and the unstable library the python code had, we were unable to have the Raspberry Pi recognize the markers we had made. Instead of using the livestreaming through the Raspberry Pi's cameras, we decided to livestream through Wen's laptop. We wouldn't be able to see the augmented trees through the robots' POV as they maneuver around, but this was the best alternative.

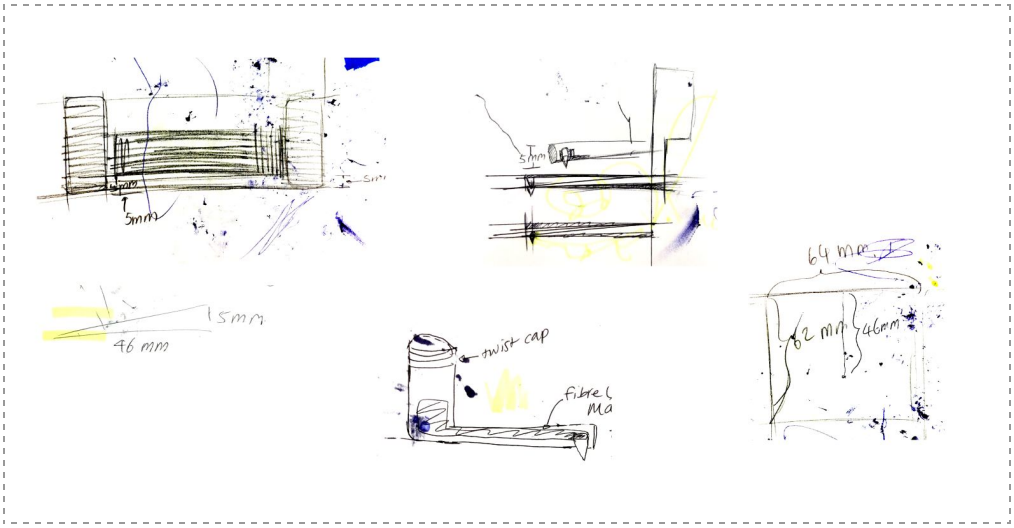


· implementation : Movement ·

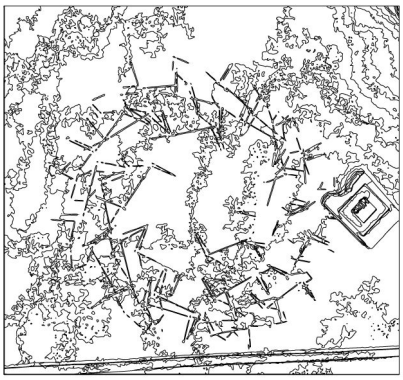
Our goal for the movement part is to program the robot to draw out constellations as they move.

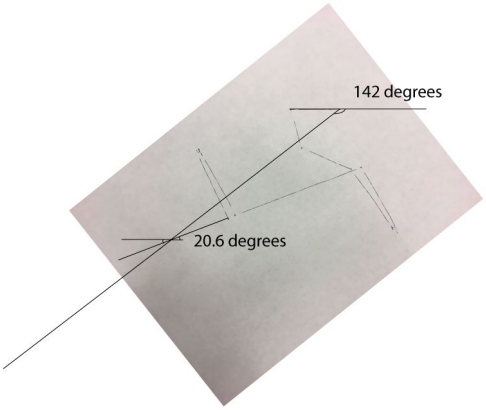


We mapped out the angles and proportions of the drawings in order to compose the Arduino commands.2

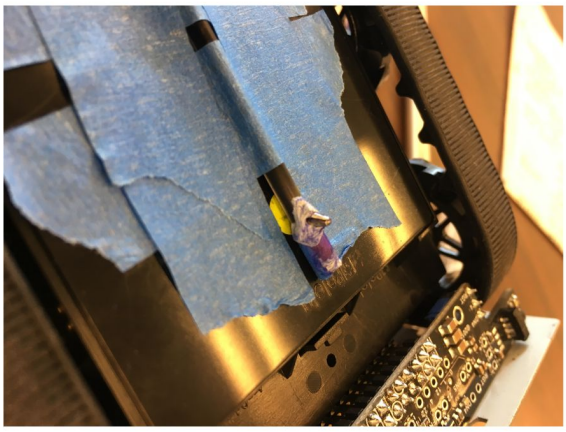


Investigations of the center of turn on the robot (right) as well as the initial designs for an ink ink.





Left: drawings in a circle before calibrating, each new dog drawing starts along the angle of the previous one's tail (imaged-traced in Adobe Illustrator since original drawing was indiscernable from paper)
Right: measurement of angles in Adobe Illustrator to make sure that each dog drawing is parallel to others.



Right: our first attempt at modifying pen refills.
Left: the "assembly line."

· code : robotic drawing movement ·

```
//a damn lot of libraries
#include <Wire.h>
//#include <Pushbutton.h>
//#include <Zumo32U4ProximitySensors.h>
#include <Zumo32U4Encoders.h>
//#include <Zumo32U4IRPulses.h>
//#include <PololuBuzzer.h>
//#include <FastGPIO.h>
#include <Zumo32U4Motors.h>
#include <LSM303.h>
#include <Zumo32U4.h>
```

```
#include <Zumo32U4LCD.h>
#include <TurnSensor.h>
//#include <Zumo32U4LineSensors.h>
#include <PololuHD44780.h>
//#include <USBPause.h>
#include <Zumo32U4Buttons.h>
#include <L3G.h>
#include <Zumo32U4Buzzer.h>
#include <QTRSensors.h>
#include <AccelStepper.h>
#include <SPI.h>
// YL: the code below is modified from
https://github.com/pvcraven/zumo_32u4_examples/blob/master/GyroSensorE
xample/GyroSensorExample.ino
// and
https://github.com/pvcraven/zumo_32u4_examples/blob/master/TurnExample
/TurnExample.ino
// and
https://github.com/pvcraven/zumo_32u4_examples/blob/master/MotorEncode
rs/MotorEncoders.ino
L3G gyro;
Zumo32U4LCD lcd;
Zumo32U4Motors motors;
Zumo32U4ButtonA buttonA;
Zumo32U4Encoders encoders;
int turnSpeed = 150;
int motorSpeed = 250;
int x = 4;
int y = 300;
int z = 100;
int i = 15;
// --- Setup Method
void setup() {
  buttonA.waitForButton();
  delay(100);
  turnSensorSetup();
  delay(500);
  turnSensorReset();
}
void turnLeft(int degrees) {
  turnSensorReset();
  motors.setSpeeds(-turnSpeed, turnSpeed);
  int angle = 0;
  do {
    delay(1);
    turnSensorUpdate();
    angle = (((int32_t)turnAngle >> 16) * 360) >> 16;
    lcd.gotoXY(0, 0);
    lcd.print(angle);
    lcd.print(" ");
  } while (angle < degrees);
  motors.setSpeeds(0, 0);
```

```
}

// Turn right
void turnRight(int degrees) {
  turnSensorReset();
  motors.setSpeeds(turnSpeed, -turnSpeed);
  int angle = 0;
  do {
    delay(1);
    turnSensorUpdate();
    angle = (((int32_t)turnAngle >> 16) * 360) >> 16;
    lcd.gotoXY(0, 0);
    lcd.print(angle);
    lcd.print(" ");
  } while (angle > -degrees);
  motors.setSpeeds(0, 0);
}
void forward(long count) {
  encoders.getCountsAndResetLeft();
  encoders.getCountsAndResetRight();
  long countsLeft = 0;
  long countsRight = 0;
  motors.setSpeeds(motorSpeed, motorSpeed);
  while(countsLeft < count) {
    countsLeft += encoders.getCountsAndResetLeft();
    countsRight += encoders.getCountsAndResetRight();
    lcd.gotoXY(0, 1);
    lcd.print(countsLeft);
    lcd.print(" ");
    delay(2);
  };
  motors.setSpeeds(0, 0);
}
void reverse(long count) {
  encoders.getCountsAndResetLeft();
  encoders.getCountsAndResetRight();
  long countsLeft = 0;
  long countsRight = 0;
  motors.setSpeeds(-motorSpeed, -motorSpeed);
  while(countsLeft < count) {
    countsLeft -= encoders.getCountsAndResetLeft();
    countsRight -= encoders.getCountsAndResetRight();
    lcd.gotoXY(0, 1);
    lcd.print(countsLeft);
    lcd.print(" ");
    delay(2);
  };
  motors.setSpeeds(0, 0);
}
void Doggo() {
   delay(y);
     // because 0.0927in
```

```
    forward(927/x);
    delay(y);
    turnLeft(96);
    delay(y);
    forward(580/x);
    delay(y);
    turnLeft(45);
    delay(y);
    forward(710/x);
    turnRight(21);
    delay(y);
    forward(1474/x);
     //should be pupper head and front limbs
    delay(y);
    turnRight(180);
    forward(1474/x);
    delay(y);
    //go back
    turnLeft(76);
    delay(y);
    forward(2899/x);
    delay(y);
    turnRight(76);
    delay(y);
    forward(1229/x);
    delay(y);
    turnRight(180);
    delay(y);
    forward(1229/x);
    delay(y);
    turnRight(93);
    delay(y);
    forward(918/x);
    //turnRight(21);
    //^this would make the new starting line parallel to the first
line of the dog, since the last line is not parallel to the first line
}
void StraightBoi(){
   Doggo();
    delay(y);
//    turnLeft(10);
//    delay(y);
    turnLeft(i);
    forward(z);
    delay(y);
}
void loop() {
  // Read the sensors
  turnSensorUpdate();
  int angle = (((int32_t)turnAngle >> 16) * 360) >> 16;
  lcd.gotoXY(0, 0);
  lcd.print(angle);
```

```
  lcd.print(" ");
  //if we press A, then it starts:
  bool buttonPress = buttonA.getSingleDebouncedPress();
  if (buttonPress) {
    while(1){
StraightBoi();
      }
}
}
```

· code : augmented reality ·

```
<!--AR.js by @jerome_etienne - github:
https://github.com/jeromeetienne/ar.js - info:
https://medium.com/arjs/augmented-reality-in-10-lines-of-html-4e193ea9fdbf
-->
<!DOCTYPE HTML>
<html>
<head>
    <title>Robo-vision</title>
    <meta name="apple-mobile-web-app-capable" content="yes">
    <meta name="apple-mobile-web-app-status-bar-style"
content="black-translucent">
    <link rel="apple-touch-icon" sizes="120x120"
href="ICON_iphone_retina.png" />
    <meta name="viewport" content="width=device-width,initial-scale=1.0,
user-scalable=no">
</head>
<script src="https://aframe.io/releases/0.8.0/aframe.min.js"></script>
<script
src="https://cdn.rawgit.com/jeromeetienne/AR.js/1.5.5/aframe/build/aframe-
ar.js"></script>
<script>THREx.ArToolkitContext.baseURL =
'https://rawgit.com/jeromeetienne/ar.js/master/three.js/'</script>
<script
src="http://class.arts.ucsb.edu/art122/Sites/s_18/esqueda_adrian/modified-
aframe-ar.js"></script>
<body style='margin : 0px; overflow: hidden;'>
    <a-scene embedded arjs='sourceType: webcam;'>
<a-assets>
          <a-asset-item id="forest"
src="forest-original.obj"></a-asset-item>
</a-assets>
        <a-marker preset="hiro">
          <a-entity obj-model="obj: #forest" scale="0.0001 0.0001 0.0001"
position:" 0 0" rotation="270 0 0"></a-entity>
          </a-marker>
    <a-assets>
          <a-asset-item id="forest2"
src="forest-original.obj"></a-asset-item>
    </a-assets>
        <a-marker preset="kanji">
```

```
            <a-entity obj-model="obj: #forest2" scale="0.0001 0.0001
0.0001" position:"0 0 0" rotation="270 0 0"></a-entity>
            </a-marker>
        <a-entity camera></a-entity>
    </a-scene>
    <style>
    </style>
</html>
```

· eventual realisation ·

The final result for our project was an interactive piece that allowed the audience to use their own devices to view the AR images. In addition, the work included a visual representation of the Greek mythology of laelaps and the Teumessian fox. Our two programmed robots performed simultaneously, drawing out constellation patterns. A poster was included for outlining our project as well as providing a QR code, which allowed our audience to participate in the AR experience using their own devices. We also included a live stream of the robots using a webcam, which exhibited our AR forests. A little boy decided that our doggo was, in fact, a good doggo, so he gave our robot many pets.

· poster ·

· distribution of labor ·

Adrian Esqueda worked mostly on the AR aspect of the project. Found and worked on the AR.js library and source code, which we relied upon to create augmented reality. He attempted to create custom made markers using Jerome Etienne's custom marker-making website, which we expected to use for the augmented reality scenes. Unfortunately, the we weren't able to create custom markers, so we resorted back to preset markers (the kanji and hiro markers). Besides working on the markers, Esqueda also worked on creating the augmented trees/forest with Unity, and added textures to the trees with Rhino. Unfortunately, none of the trees textures or colors translated well when he turned them into OBJ files.

Wen Ying Liu, like Adrian, worked on the AR aspect of the project. Attempted to create custom markers using Jerome Etienne's custom marker-generating website, but it didn't work. She also designed a tree using Unity, using YouTube tutorials to create a realistic looking tree. She was, unfortunately, unable to convert the tree she rendered into an OBJ file, so she left that task to Esqueda. Wen was tasked on making the Raspberry Pi's webcam to livestream, using the code Rodger Luo wrote. She spent weeks attempting to connect her computer to the Raspberry Pi's browser and camera, and while she was successful in getting the Raspberry Pi's webcam to livestream real time, the Raspberry Pi's cameras were unable to scan the markers. We initially wanted to see the augmented trees through the Raspberry Pi's POV (which would have been projected onto the big screen). Since the camera framerate was too weak, and thus weren't able to recognize the markers, we decided to use Wen's laptop's webcam to livestream the robots' environment.

Lawrence Chen worked on the robots' movements, AR and model creation. Lawrence designed the poster and acquired materials such as paper and additional materials for the project. He also designed the original movement code, which would result in a pattern resembling a 4/4 conducting gesture, as well as the initial plan for an ink tank. He often provided transportation for the rest of the group to classes and to meetings.

Yichen Li worked on the robots' movements and drawing code. She attempted to design an ink tank system along with Lawrence. However, after researching types of pens and inks, it was concluded that the only solution for using oil-based ink was modifying pen refills. Along with Lawrence, Yichen tested out the initial movement code. The initial movement code had to be abandoned since its errors were significant. Yichen also tried attaching textures to obj. files on Rhinoceros, which failed to import to the final website.

Lawrence and *Yichen* also attempted using custom AR markers by modifying the ar.js library as suggested by an online work-around.

Emily Beckius provided assistance for the Ar system, including testing functions, creating models, and printing markers. She also helped provide obj. Files. In addition, she had attention to detail and assisted in the creation of the poster as well as set up.

· references ·

https://github.com/jeromeetienne/AR.js/blob/master/README.md

https://github.com/RodgerLuo/robotic-vision/tree/master/streaming_python

https://github.com/pvcraven/zumo_32u4_examples/blob/master/GyroSensorExample/GyroSensorExample.ino

https://github.com/pvcraven/zumo_32u4_examples/blob/master/TurnExample/TurnExample.ino

https://github.com/pvcraven/zumo_32u4_examples/blob/master/GyroSensorExample/TurnSensor.cpp

https://github.com/pvcraven/zumo_32u4_examples/blob/master/MotorEncoders/MotorEncoders.ino

https://gizmodo.com/whats-the-ink-in-a-standard-rollerball-pen-made-of-1466794448

https://katharine.org/tutorials/custom-markers-ar-js/