STARBUCKS STEVE: A GESTURE-BASED COFFEE COUNTER ASSISTANT

Ananya Nandy, Sophia Batchelor, Will Oakley, Jessie Lyu

Intro

Steve is a smart coffee assistant that helps customers make decisions about their drinks. Upon perceiving a customer, Steven will prompt a series of binary choice questions such as flavor, caffeine, or milk choices. Users interact with Steve with hand gestures like pointing left or right. Steve responds by shaking his left or right “arms”, and gives feedback on both the LCD screen and through a speaker. Designed in a shape that resembles a real Starbucks coffee cup, we hope Steve could help users make beverage choices more easily!

Design Process

Brainstorming and observation

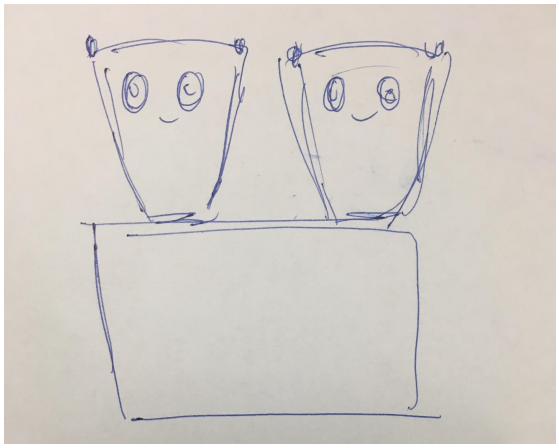
We started our project by brainstorming ideas regarding the “counter” table. We listed a few common contexts where counter tables are important such as kitchen, retail stores, and restaurants. Each of us started to talk about our experience with the counter table, and we found all of us were interested in the counter table in coffee stores. As students, we all need to go to coffee shops to get a cup of latte or Americano to refresh ourselves. Designing for a coffee shop could be an interesting project since a lot of interactions happen in front of the coffee counters.

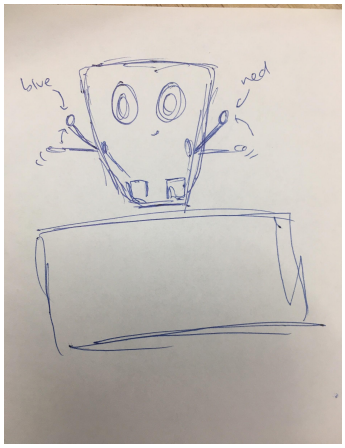
After narrowing down our context, we began to brainstorm specific interactions to focus. We went to real coffee shops to observe how customers order drinks and found out possible ways to improve their experience. During the observation, we found that most stores have very long menus. However, what a customer needs might be as simple as “something that is ice and caffeinated without milk”, which suggests a cup of Americano. The word “Americano” on the menu might not be intuitive enough for customers to pick up the drink within seconds, so many customers spend minutes staring at the menu without knowing what exactly they are going to order. This is very inefficient since it will cause a longer wait time for the following customers in line.

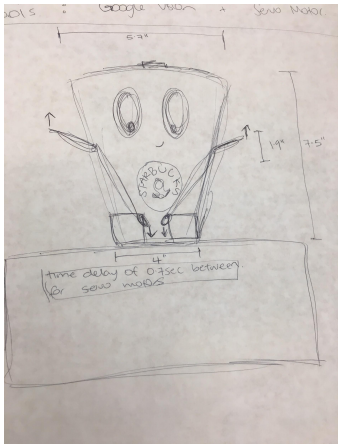
We thought the ordering process could be expedited if we breakdown coffee decisions into several binary choices questions, like ice or cold, caf or decaf. In addition, this could reduce the effect of choice overload from huge menus. We thought this could be achieved by an autonomous coffee recommendation machine. The machine will be equipped with a camera which could perceive the actions of the customer, and streamline the coffee purchase process by prompting out binary choice questions. The customer could respond to the questions with gestures, and the machine will have visual, audio and motor output to confirm customers’ choices. See Appendix A for observation notes.

Ideation

When we were designing the look of the coffee recommendation machine, we got inspirations from Starbucks coffee cups. Since we are designing a “smart” bot, we thought it would be better to humanize the cup to make it like a live assistant instead of a static machine. We added “eyes” to the cup to make it resemble a human. We gave the name “Steve” to the machine because it is a common name and can seem friendly.







Initially, we thought about using two coffee cups as metaphors for binary choices. The cups stand for choice 1 and choice 2, and whenever the customer picks a choice, the cup on that side will be lifted by the motor. However, we felt this design is not intuitive since customers normally interpret different cups as different drink sizes. So in later design, we changed it to a single cup with two “arms”. The arms stand for the two choices, and whenever the customer picked a choice, the corresponding arm will be lifted higher by the motor.

Interaction design

We designed the interactions between Steve and customers in several aspects. Since Steve will be equipped with a camera, we used the camera as the major input source of Steve. The camera will capture customer gestures, and use Google Vision API to label each image. We wrote an algorithm to categorize each image to either choice A or choice B according to the user movement by using object localization given by Google Vision API.

Steve had three types of output: motor, LCD screen, and audio. As mentioned above, the motor will lift the “arms” of Steve to respond to the customer’s inputs. For example, if the customer picks the right side choice, the right arm will be lifted to confirm the choice. The LCD screen is used to give the customer questions and guide his choices. For example, if the question is about coffee temperature, the screen will display “hot” and “cold” as the two choices. The audio is used as cues of activation or confirmation. For example, when Steve is triggered to start, a sound will be played through the speaker.

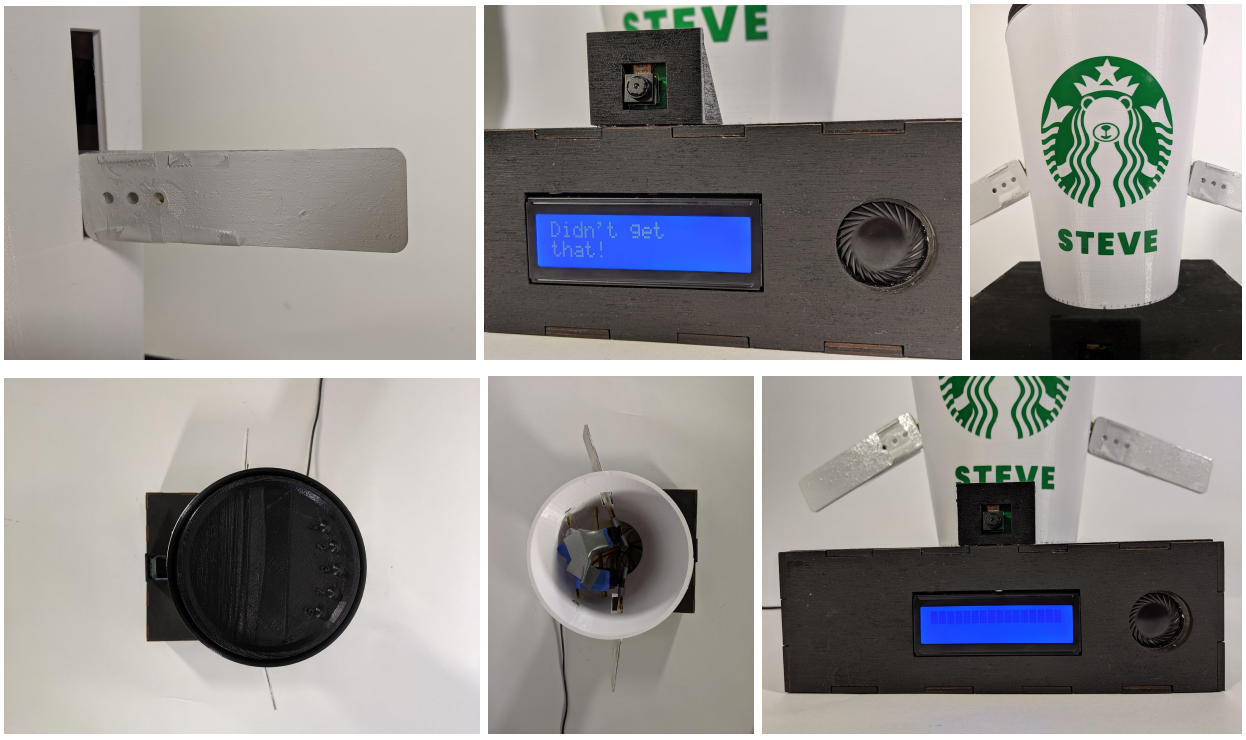
A typical user flow of Steve will be similar to the following:
A customer enters the coffee store such as Starbucks, he stands in front of the counter table to go over the menu and is overwhelmed by the many choices. After 15 seconds (our demo cuts this down to 5 for convenience), Steve is activated by playing a sound. A welcome message is prompted on the LCD screen followed by the first question, hot or cold. The customer points right to indicate desire for an iced drink. The right arm is lifted to confirm the choice. Then the second question is prompted asking about coffee or tea. Users will be asked several questions following

the same manner regarding drink temperature, type, caffeine, etc. Finally, using the user's inputs, Steve outputs a drink recommendation for the customer to help make the drink decision.

Step-by-Step Building Guide

Fabrication

We decided to 3D print Steve's body. Since our inspirations were from Starbucks, we decided to print the cup as similar to the Starbucks coffee cup as possible. So we chose white PLA material as the cup body and the black PLA material as the cup lid. We laser-cut the arms of Steve and painted it in white to match the color of Steve's body. We designed a simple mechanical structure for the arms and attached them to two motors. Every time the motor rotates, the mechanical structure will lift the arm up or down. We also laser-cut the bottom box stand to hold Steve. The stand was painted in black with all the electronics inside.



Besides the basic fabrication, we created a logo for Steve. Keep in mind that Steve is designed an "assistant" in Starbucks, we customized the original Starbucks logo to create a "Stevebucks" logo, by replacing the mermaid head with a bear head[1]. We vinyl printed the logo and stuck it onto the body of Steve.

[1] Go Bears!

Electronics

List of electronics we used:

- A Raspberry Pi
- A Crickit
- A speaker
- Two servo motors
- A LCD screen
- Wires

Code

Steve's codebase has 4 parts.

- Core program
- Drink selection
- Movement control
- LCD display

The 4 programs integrate throughout to form his functionality.

Steve's core program involves using Google Cloud's Vision API as a triggering event and to determine what the subsequent function that runs will be.

We built the code so that it would take two photos and compare the contents for a person.
If a person was detected by Steve and then still detected after a certain amount of time, Steve would activate.
We took standing in the same position pondering the coffee menu as the trigger for the person needing some help deciding what to order.

```
takephoto(camera)
with open('image.jpg', 'rb') as image_file:
    #read the image file
    content = image_file.read()
    #convert the image file to a GCP Vision-friendly type
    image = vision.types.Image(content=content)
    first = person_seen(image)
time.sleep(2)
takephoto(camera)
with open('image.jpg', 'rb') as image_file:
    content = image_file.read()
    #convert the image file to a GCP Vision-friendly type
    image = vision.types.Image(content=content)
    second = person_seen(image)
if first and second == True:
    seen = True
else: seen = False
```

Upon activating (if the person was detected as having stood in front of the camera for some time), Steve would introduce himself via the LCD screen, state his purpose, and begin his posturing.

```
while seen == True:
    #play the on sound
    #pg.mixer.music.load(on_sound)
    #pg.mixer.music.play()
    lcd = LCD()
    lcd.showCustomMessage("Hi! I'm Steve!")
    satcode.steve_sleeping()
    time.sleep(2)
    play_on_sound()
    satcode.steve_sees_you()
    lcd.showCustomMessage("I'm here to help") #this is how you get screen print
        outs
    questions = [("Hot", "Cold"), ("Coffee", "Tea"), ("Caff.","Decaf"),
        ("Flavor", "Plain")]
    answers = []
```

We chose a friendly, electronic sound that could be interpreted as a “hello” to indicate that Steve was in an active state. The servo motors were one of the more finessed parts of the code (compared to the LCD messages and drink options which were programed to pair with their indicated properties).

Steve in an “Off” position had his arms completely lowered while “Awake” has different gradations based on the interaction.

The way Steve could detect what answer/selection you made was based on the images taken by the camera. The images Steve took in the beginning of the sequence were used to determine the center of the bounding box around what Google’s Vision API determined was a “Person”. This was the “initial” position. In addition, we determined the “active” person to be the person with the biggest bounding box to avoid interference by people who were behind you in line.

When there was an expansion of the bounding box following a “question” from Steve, the direction of expansion was marked and taken in as the “answer”. Therefore, if the bounding box expanded in one direction due to the person’s gesture, it was recorded as a corresponding answer. Initially, we reset the “initial” bounding box center between each question but this slowed the interaction down significantly so we decided that the “initial” would just be set from the activation. This meant we were assuming that the person would not significantly move (other than gesturing) between the choices.

```
#print('Number of objects found: {}'.format(len(objects)))
for object_ in objects:
    #print('\n{} (confidence: {})'.format(object_.name, object_.score))
    area = 0
    if object_.name == 'Person':
        vertices = object_.bounding_poly.normalized_vertices
        #print(vertices) #counterclockwise from bottom left
        length = abs(vertices[0].x - vertices[1].x)
        height = abs(vertices[1].y - vertices[2].y)
        new_area = length*height
        if new_area > area:
            area = new_area
            center_x = vertices[0].x + (length/2)
            center_y = vertices[0].y + (height/2)
            return center_x,center_y
            print(center_x, center_y)
```

This was then mapped in our “decision_maker” function which determined if a “right” or “left” selection was made and returned the corresponding choice (i.e. Hot) on the LCD screen

```
def decision_tracker(chosen, question):
    if chosen == "left":
        return question[1]
    elif chosen == "right":
        return question[0]
    else:
        return "none"
```

There was initial confusion when creating and setting these rights and lefts and as a group we decided that R/L would be decided from Steve’s perspective for movement.

```
  if decision == "right":
      lcd.showCustomMessage(q[0])
  elif decision == "left":
      lcd.showCustomMessage(q[1])
  else:
      lcd.showCustomMessage("Didn't get\nthat!")
  #(oldx, oldy) = (x,y)
  seen = False
  time.sleep(3)
```

Steve’s arms were designed to mimic a human like motion of offering two options (ie. the traditional right and left hand presentation). With a “lift” motion that could be understood by the user as Steve confirming that he received their input.
Steve therefore has 4 gesture settings (off, active, offering options, and confirmed option) which would be triggered by specific user input.

```
def steve_sleeping():
    #servos = (crickit.servo_1, crickit.servo_3)
    print ("I see nothing")
    servos[1].angle = 60
    servos[0].angle = 0
    time.sleep(2)

#once detected a face
def steve_sees_you():
    #servos = (crickit.servo_1, crickit.servo_3)
    print ("about to move arms")
    servos[1].angle = 40
    servos[0].angle = 20
    time.sleep(0.5)

def steve_gives_options():
    #servos = (crickit.servo_1, crickit.servo_3)
    print ("offering options")
    servos[1].angle = 40
    servos[0].angle = 20
    time.sleep(1)
    servos[1].angle = 10
    time.sleep(0.5)
    servos[0].angle = 40
    time.sleep(0.2)


def steve_recieved_your_choice(decision):
   # servos = (crickit.servo_1, crickit.servo_3)
    if decision == "right":
        print ("steve's right option was chosen")
        print("i'm moving my right hand")
        servos[1].angle = 0
        servos[0].angle = 0
        time.sleep(1)
    elif decision == "left":
        print ("steve's left option was chosen")
        print("i'm moving my left hand")
        servos[1].angle = 60
        servos[0].angle = 60
        time.sleep(1)
```

(We named our functions very explicitly.)

The role of Steve was to make your drink decision easier and to break down the choice overload so we built out a drink selection which mapped some commonly ordered drinks to Steve's options. An intersection was used to get a drink that fit with the user's inputs The randomizer would then output a selection that matched and then the drink recommendation would print out on the screen. If the user input for any of the questions was "None" (they didn't make a decision about that question), that input was not incorporated in the choice of drink.

```
random.seed(time.clock())

drinks = {
    "Hot":["Tea","Coffee","Herbal Tea","Mint Tea", "Green Tea", "Hot Choc",
        "Mocha", "Chai Latte", "Dirty Chai", "Decaf Coffee"],
    "Cold":["Iced Tea","Iced Coffee","Iced Herbal Tea", "Iced Black Tea", "Iced
        Chai Latte", "Iced Peach Tea", "Iced Mocha", "Flavored Frap", "Coffee
        Frap", "Iced Choc"],
    "Coffee":["Iced Coffee","Coffee", "Mocha", "Dirty Chai", "Iced Mocha", "Coffee
        Frap", "Decaf Coffee"],
    "Tea":["Tea","Iced Tea","Herbal Tea","Iced Herbal Tea","Mint Tea", "Green Tea",
        "Iced Green Tea", "Iced Peach Tea", "Black Tea", "Iced Black Tea"],
    "Decaf":["Mint Tea","Iced Herbal Tea","Herbal Tea","Mint Tea", "Chai Latte",
        "Iced Chai Latte", "Iced Choc", "Hot Choc", "Iced Peach Tea"],
    "Caff.":["Tea","Iced Tea","Iced Green Tea", "Iced Black Tea", "Dirty Chai",
        "Iced Dirty Chai", "Flavored Frap", "Coffee","Iced Coffee", "Coffee Frap",
        "Mocha", "Iced Mocha"],
    "Flavor":["Hot Choc", "Chai Latte", "Dirty Chai", "Iced Chai Latte", "Iced
        Peach Tea", "Iced Mocha", "Mocha", "Flavored Frap", "Mint Tea", "Iced Green
        Tea", "Iced Black Tea"],
    "Plain":["Coffee Frap", "Iced Coffee", "Coffee", "Tea", "Iced Tea", "Iced Green
        Tea"]
    }

def your_order(answers):
    drink_result = []
    for answer in answers:
        drink_result.append(drinks[answer])

    drink_result = set.intersection(*map(set,drink_result))

    return random.choice((list((drink_result))))
```

And that's how Steve helps you order at a coffee shop!

Appendix A - Observation Notes

Starbucks

We went to Starbucks to observe the food ordering. In Starbucks, we found there are 8 pages of menus, with 6 pages for promotions and new items. We also noticed that the menu is located on the back of the counter so customers can read the menu only if they are close enough.

Yali's Cafe in Sutardja Dai Hall
During peak hours, Yali's Cafe is very crowded, and normally there are more than 10 people waiting. So typically people come to the counter knowing what they want to order already. Yali's Cafe has a huge menu at the waiting line area, which is a smart design decision. We observed that many customers look through the menu very carefully while waiting.

Appendix B - Files

https://github.com/ananyan/counter-culture/