H13.
Human Interaction with an Avatar

Visualized character that demonstrates emotions by reading facial gestures.
Blueprints for Natural Language and Machine Learning to interact with avatar.

Tyler Hull | **Intelligent Robotics SPR 2020** | **Project**

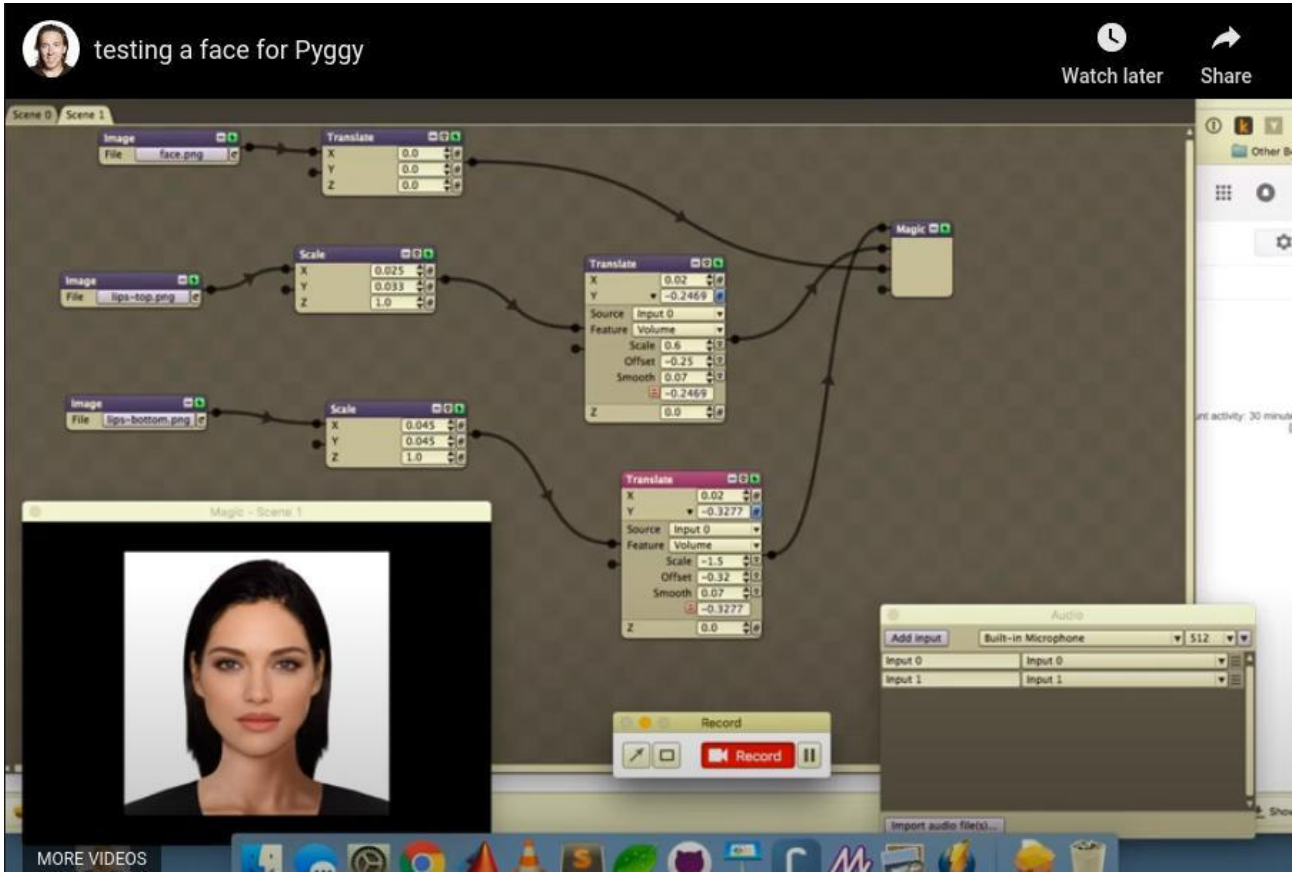
Software Used in Project

- Adobe Illustrator (Creating Vector art for Avatar)
- Adobe Photoshop (Editing Avatar Art and Generating Images)
- Adobe Character Animator (Some animations and rigging)
- Python 3.8 (Python 3.8 for GUI and OpenCV control)
- OpenCV 4.2 (Used for Machine Learning and video app)
- wxPython (Cross Platform GUI for Python)
- Amazon LEX (used for building conversational interface)

Research into Controlling Animations with Python

Looking into Improv

Kory Mattheson, Research Scientist, Google (DeepMind)
https://github.com/korymath/Pyggy



Magic
Music Visuals

Home | Download | Purchase | Features | Gallery | Support | Forums

Music Visualizer, VJ Software & Beyond

Fully-customizable and highly responsive visuals for live and recorded audio & MIDI.

Download the free Demo now for Mac or PC.

News

• Magic v2.22 has been released! This version has some great new features, including a new Iterator module, several new sample projects, and a few fixes/improvements. **View the announcement** on our forums, or just **download the free Demo now**!

pandorabots Resources Features Leadership Services Pricing Sign In

Chatbots for **marketing**

Message-enable your business using the leading conversational AI platform

Sign Up Free

275,000+ REGISTERED DEVELOPERS

325,000+ </> CHATBOTS CREATED

75,000,000,000+ MESSAGES PROCESSED

Legacy Pandorabots API Docs Sign In

AIaaS

Build conversational AI for applications.

LEARN MORE SIGN IN

Web Speech API Demonstration

Does the web speech API

Copy and Paste Create Email

English United States

PYGGY

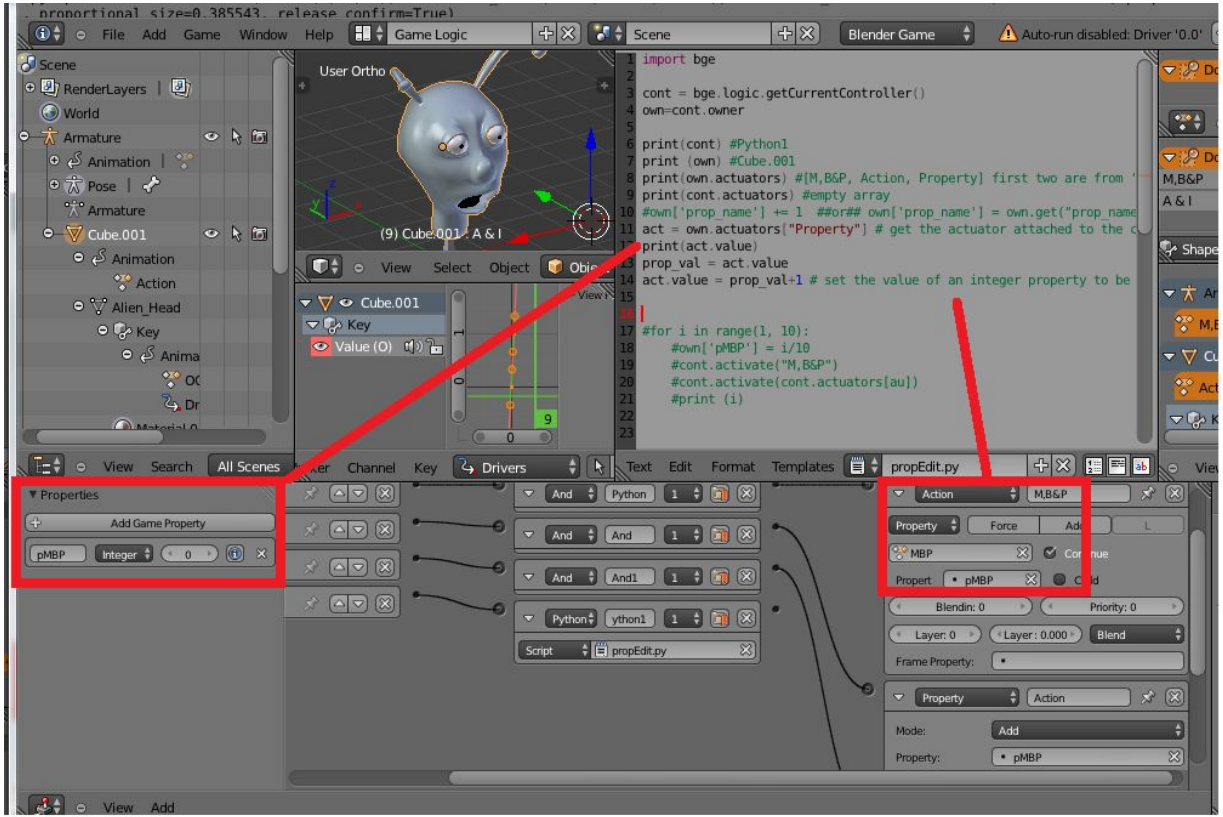
Pros

- Open source
- Conversational chatbot working
- Already contains avatar images

Cons

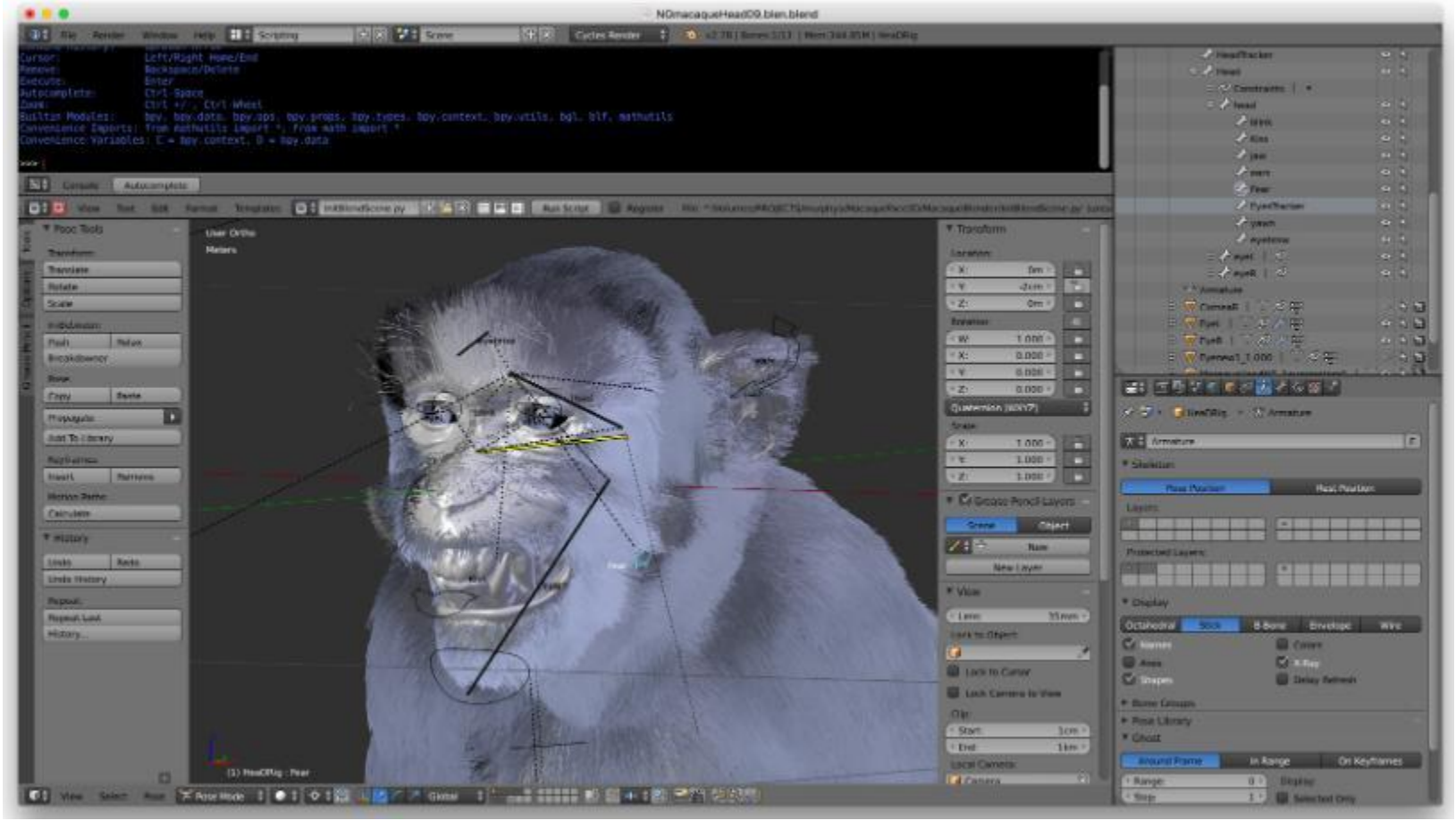
- Avatar is very limited and uses DJ software for animation
- Uses PandoraBots and Artificial Intelligence Markup Language (AIML)
- Project is older and used some early API versions

Scripting In Blender

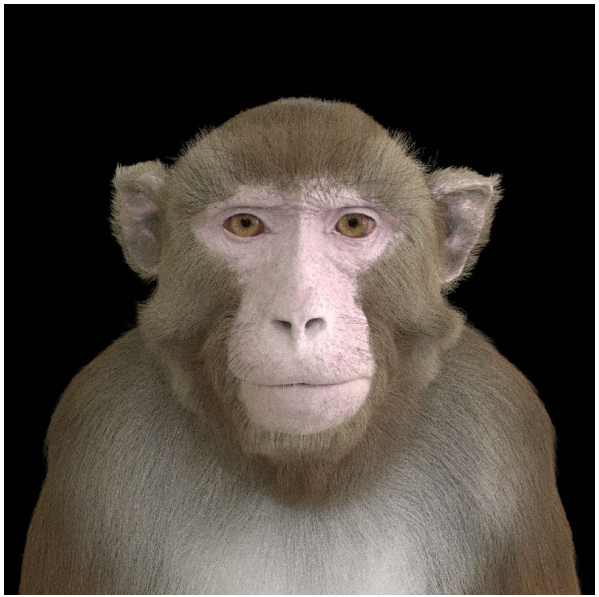


Adian Murphy, Primate Neuroscientist, Princeton

https://github.com/MonkeyGone2Heaven/MacaqueBlender/blob/master/README.md



Animations created with MacaqueBlender project



Blender

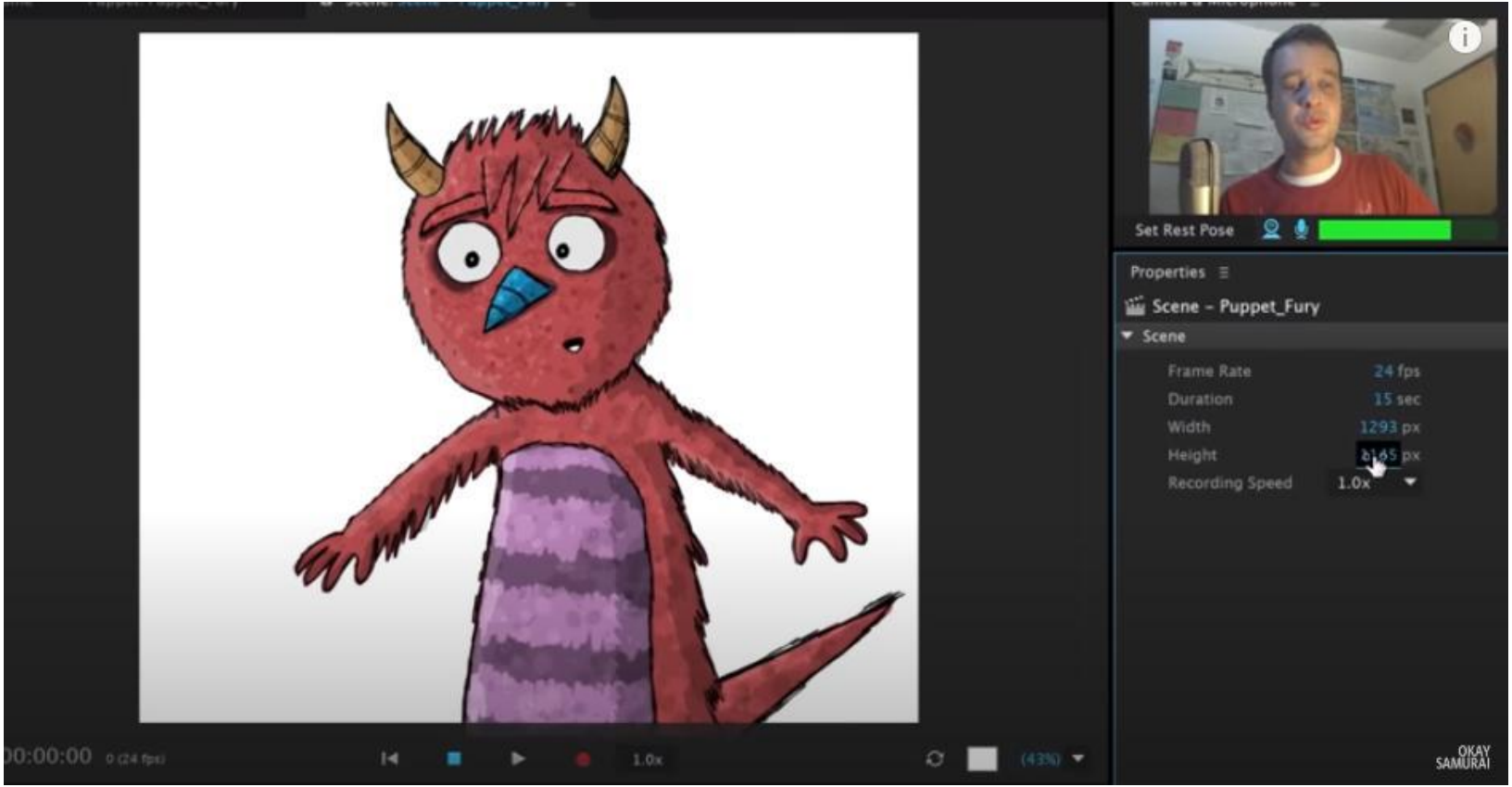
Pros

- Open source animation and 3D software package
- Lots of community support and tutorials
- Python scripting built in for creation of animations or control

Cons

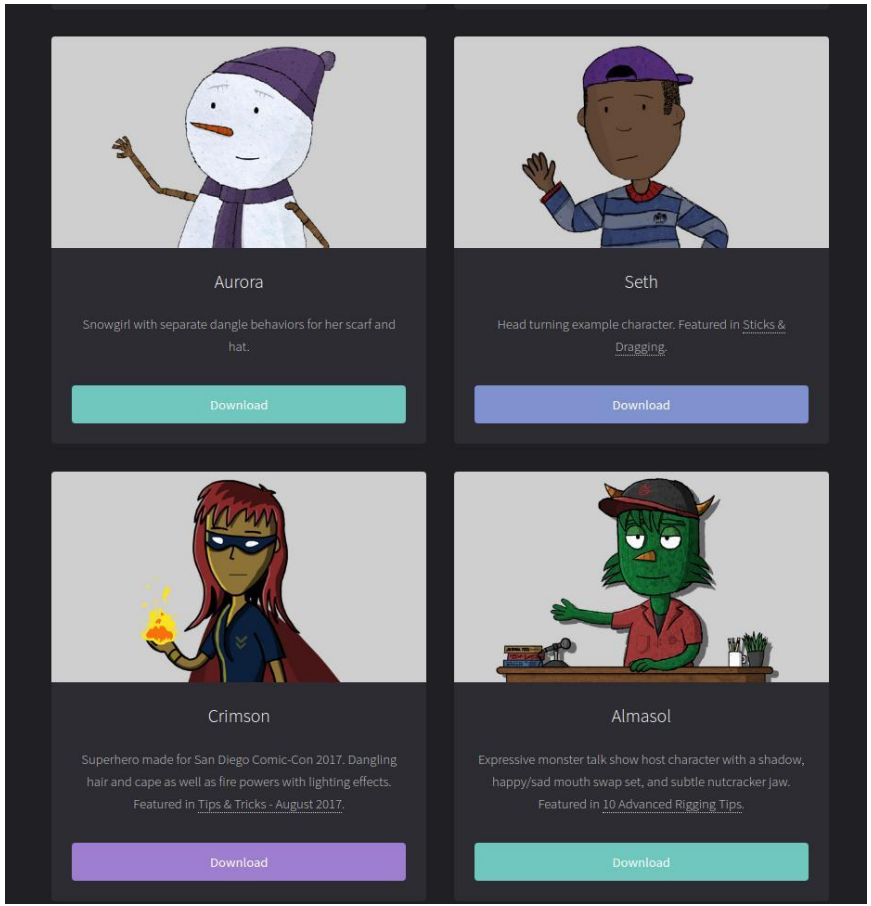
- Software is complicated and was unfamiliar to me
- Creating and rigging 3D animations is challenging
- Not a lot of information for controlling fully rigged models with Python

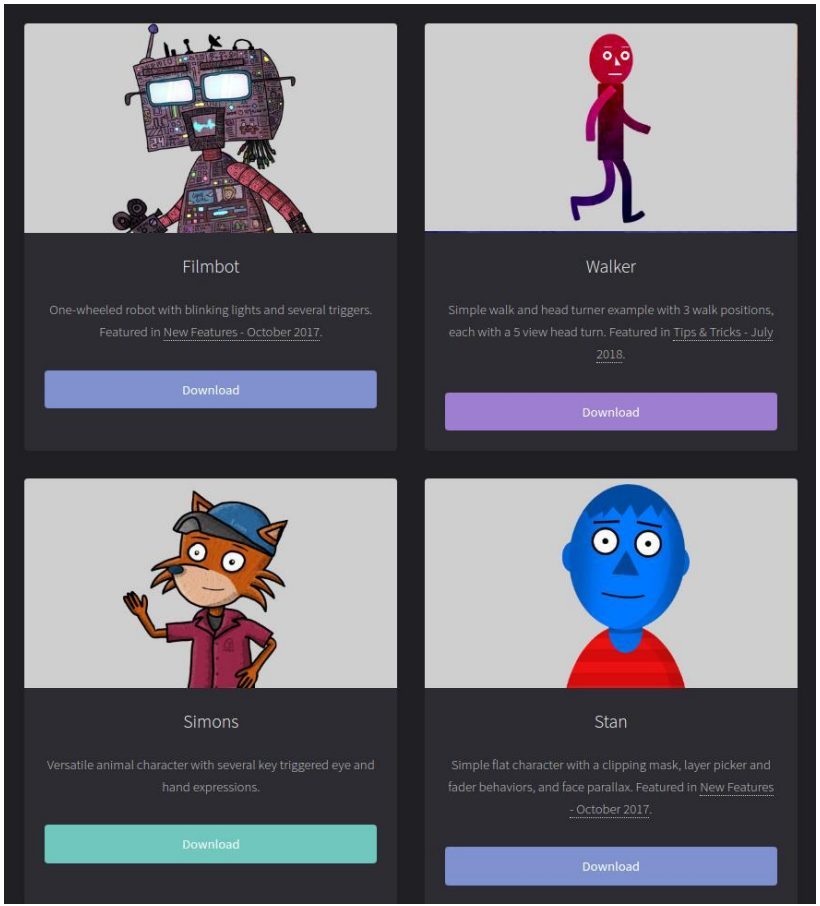
Avatar with Adobe Character Animator


Set Rest Pose
Properties
Scene – Puppet_Fury
Scene
Frame Rate 24 fps
Duration 15 sec
Width 1293 px
Height 1165 px
Recording Speed 1.0x
00:00:00
1.0x
(43%)
OKAY SAMURAI

Lots of Pre-Rigged “puppets” and tutorials Available

https://okaysamurai.com/puppets/





Character Animator

Pros

- Lots of tutorials available online
- Lots of premade Avatars or “Puppets” available for download
- Familiar tools that were similar to Adobe After Effects and Illustrator/Photoshop

Cons

- Closed Source and has monthly cost ($25/month for students - All adobe Apps)
- Needs to output a video or frame by frame images for use in OpenCV
- No Scripting built in, so it does internally what we want, but no software control

Decided to go with Adobe Character Animator

- Thought this would simpler based on my experience with Adobe
- Misunderstood the complexities of trying to export and match up video clips
- At the time, I wasn’t aware that there was no scripting support available yet

Think Like Disney

Or Jim Henson

Build the Avatar, so it Can also be the Real Thing.

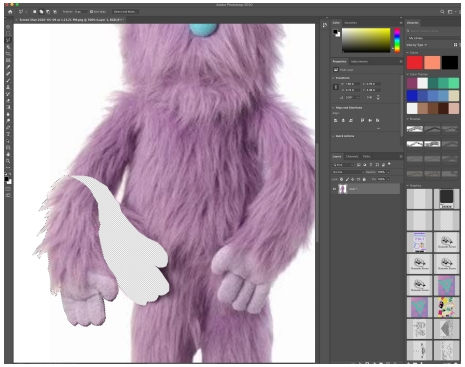
- No Devil Character (Or other scary characters)
- No copyrighted characters (In the spirit of disney, not copied from disney)
- Characters with Fur don’t require as precise of movements
- Eye movements and Eye Brows, really important for emotions animation.
- Next year, students working on robot heads

Working Backwards from a puppet

https://www.puppetsandprops.com/wp-content/uploads/2018/05/The-Fred-Project.pdf

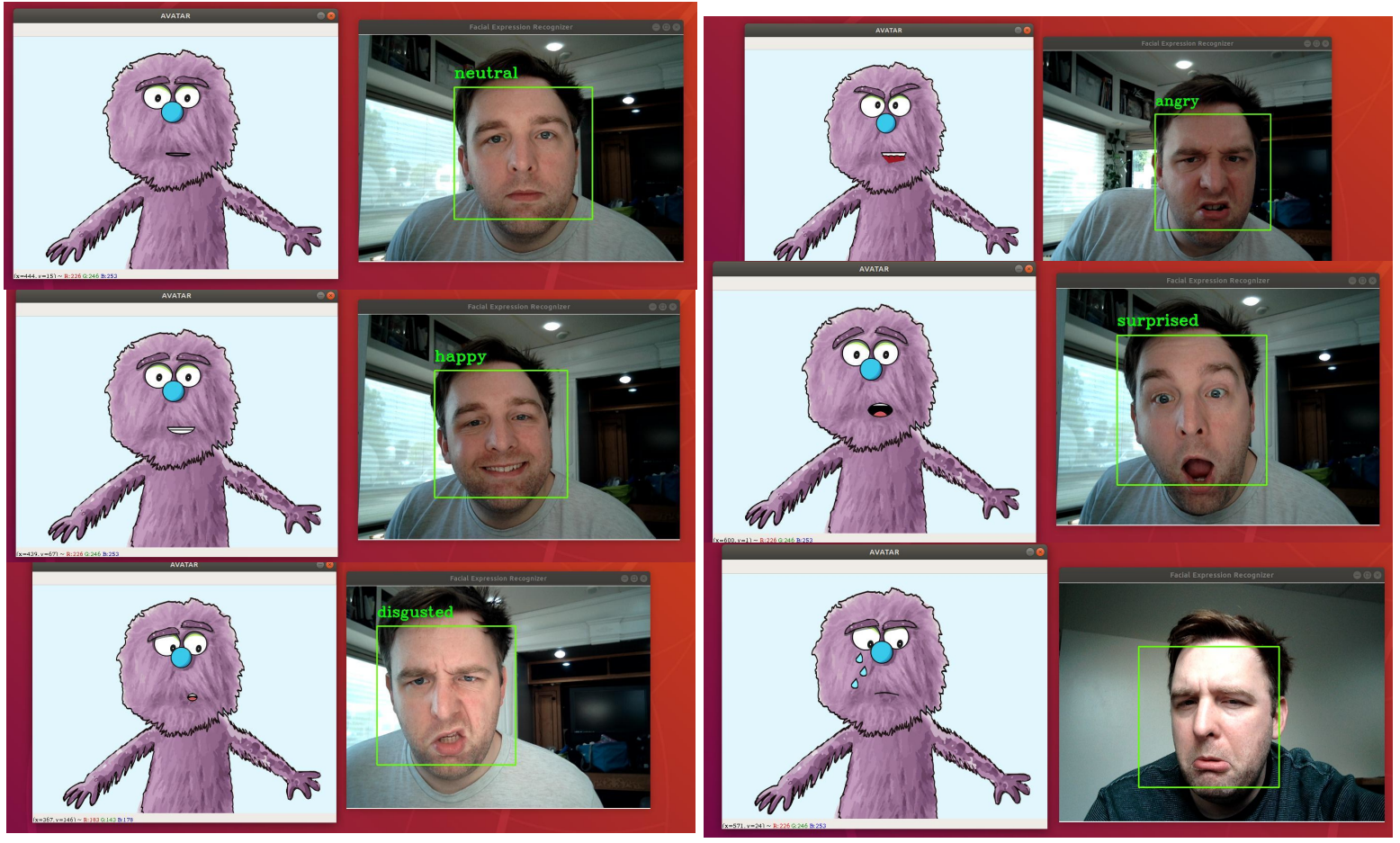
https://www.amazon.com/Purple-Monster-Puppet-Ventriloquist-Style/dp/B004FRZRL6/ref=sr_1_37?dchild=1&keywords=Silly+Puppets&qid=1592098345&sr=8-37

Working Backwards from a puppet



Ella in Adobe Character Animator

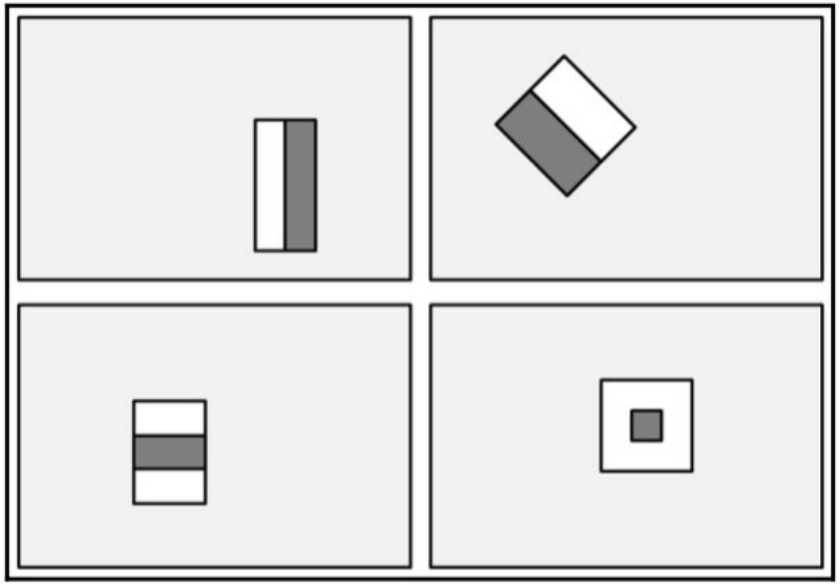
Reading and Displaying Emotions



Avatar Emotions App Overview

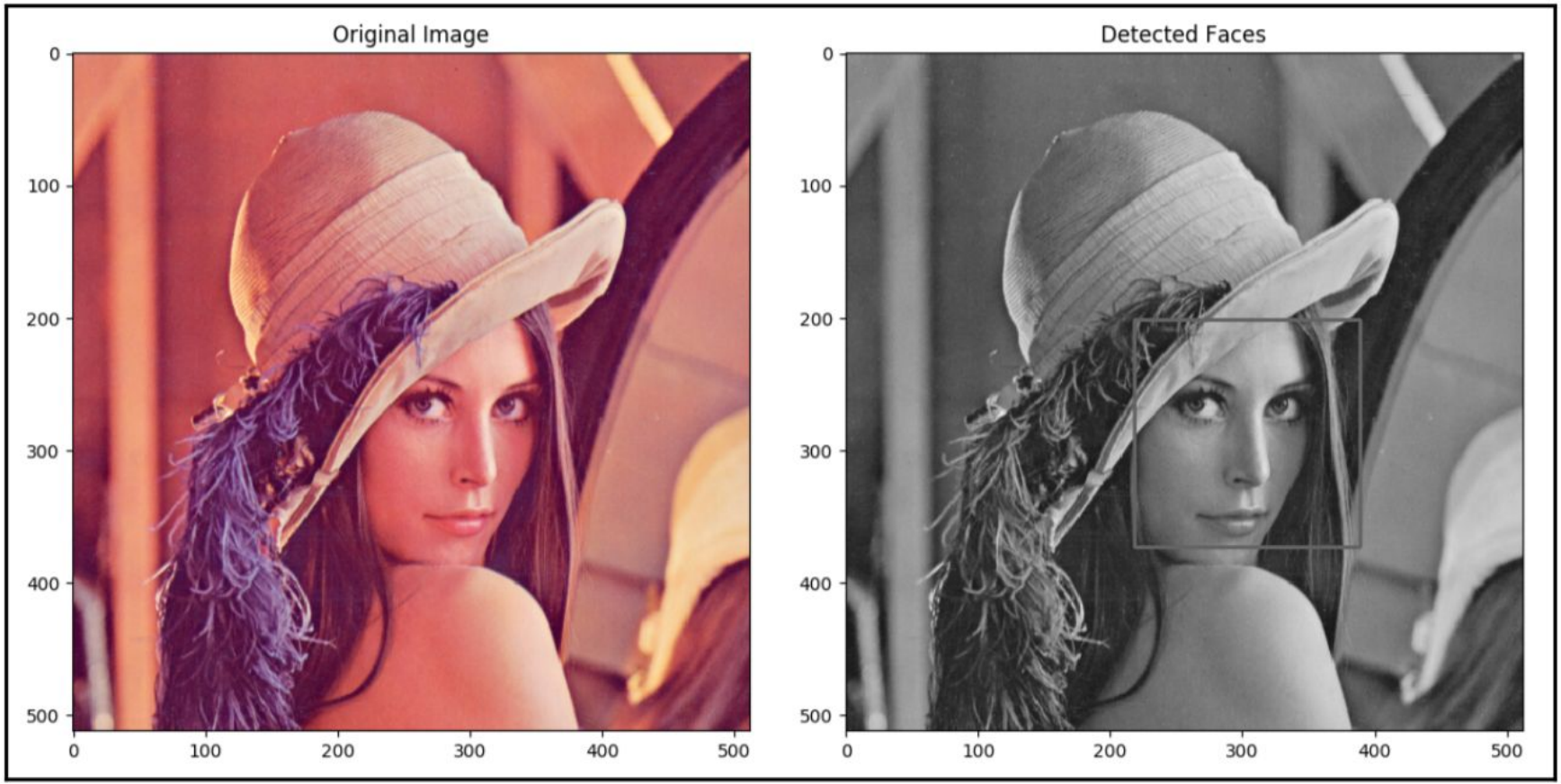
- Uses OpenCV built in Haar cascade classifiers (Viola-Jones)
- Uses OpenCV built in Multi-level Perceptrons (MLPs)
- Uses Principal Component Analysis (PCA) to reduce dimensionality of the data and speedup algorithm accuracy
- Setup to read 6 emotions, but could be expanded. (Neutral, happy, sad, angry, surprised, disgusted)

Haar Cascade Classifiers in OpenCV



- Detects differences between light and dark areas in grayscale images
- Many different options built into OpenCV for different features
- For this project we're using haarcascade_frontalface_default.xml and haarcascade_eye.xml
- Need to convert our image to grayscale before we can run.

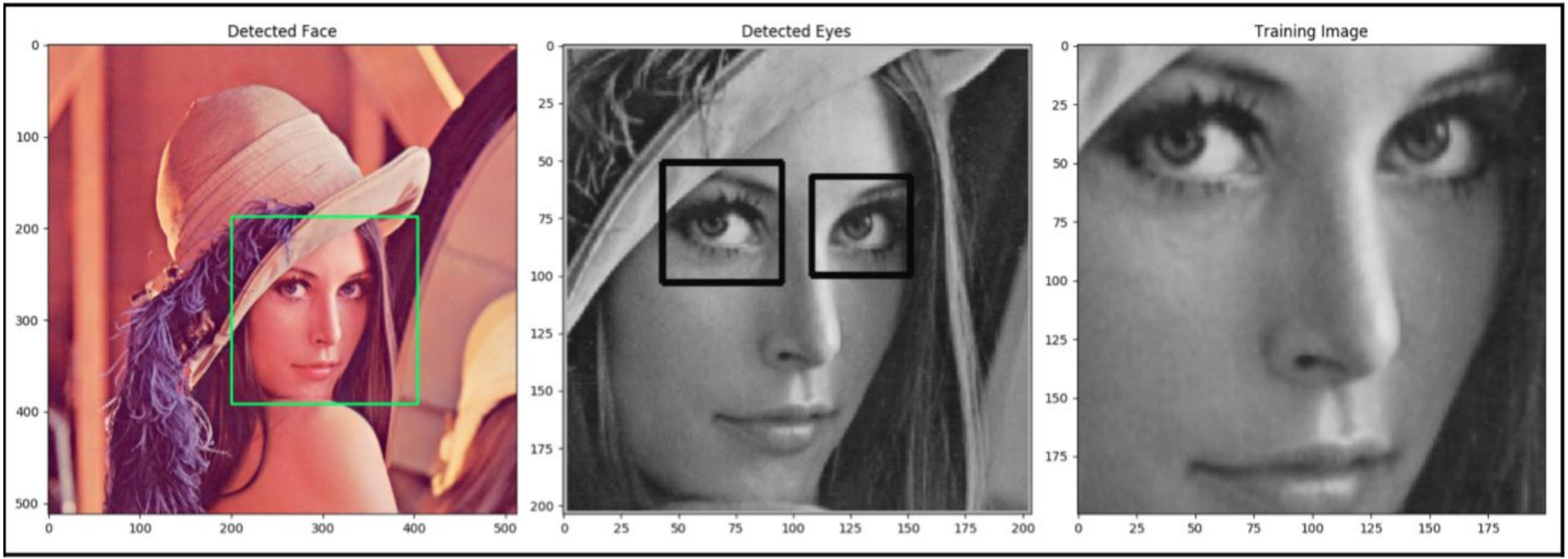
Detect a Face & Color Gray



Detect a Face & Color Gray

```
import cv2

gray_img = cv2.cvtColor(cv2.imread('example.png'), cv2.COLOR_RGB2GRAY)

cascade_clf = cv2.CascadeClassifier('haarcascade_frontalface_default.xml')
faces = cascade_clf.detectMultiScale(gray_img,
                                     scaleFactor=1.1,
                                     minNeighbors=3,
                                     flags=cv2.CASCADE_SCALE_IMAGE)
```

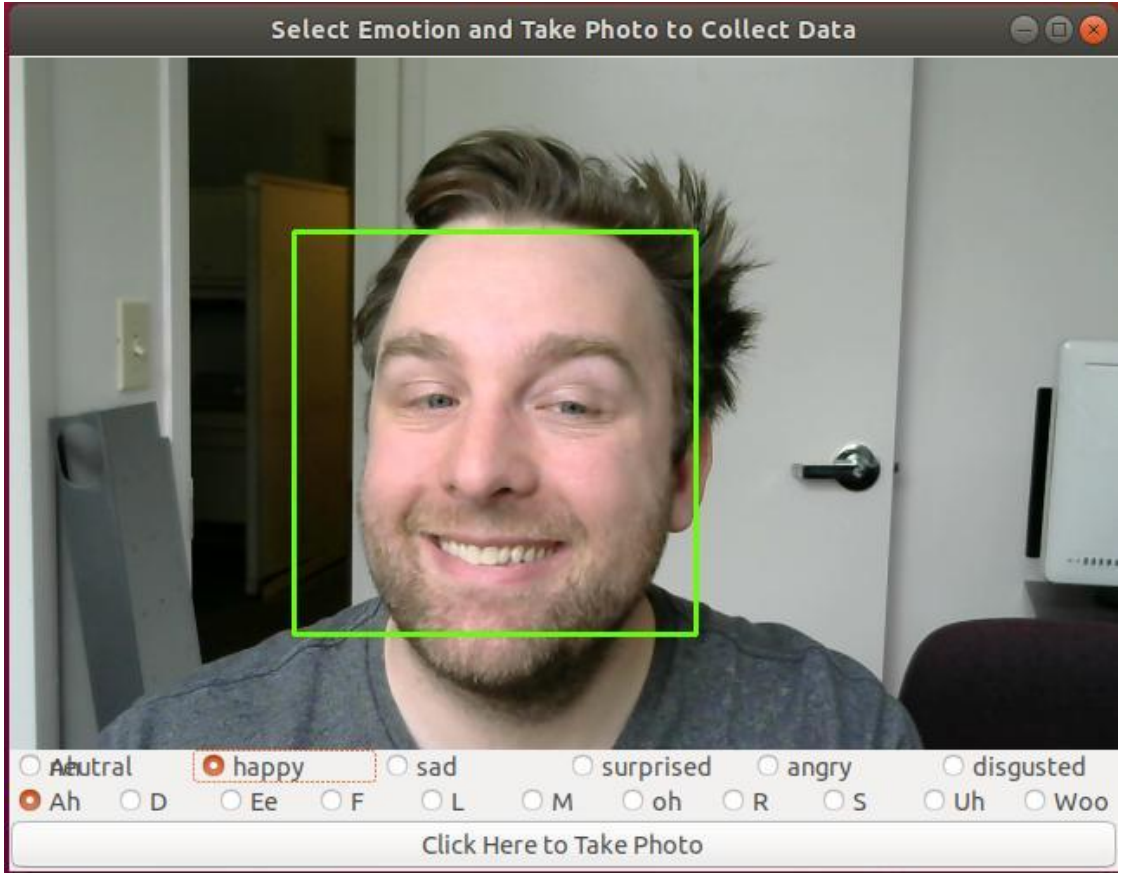
Detect Eyes & Transform Face



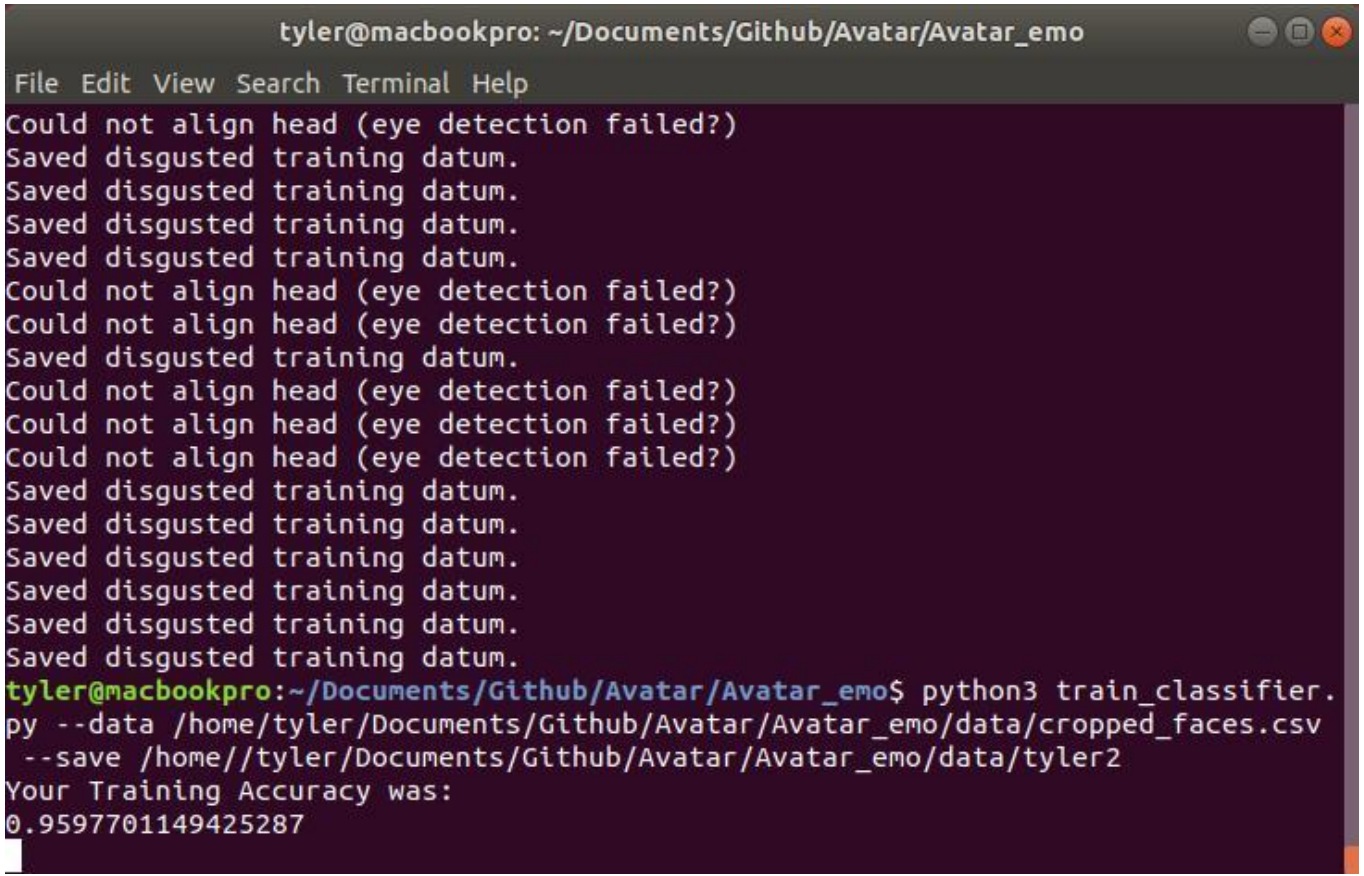
Detect Eyes & Transform Face

```
# we want the eye to be at 25% of the width, and 20% of the height
# resulting image should be square (desired_img_width,
# desired_img_height)
desired_eye_x = 0.25
desired_eye_y = 0.2
desired_img_width = desired_img_height = 200

try:
    eye_centers = self.eye_centers(head)
except RuntimeError:
    return False, head

if eye_centers[0][0] < eye_centers[0][1]:
    left_eye, right_eye = eye_centers
else:
    right_eye, left_eye = eye_centers

# scale the distance between eyes to desired length
eye_dist = np.linalg.norm(left_eye - right_eye)
eyeSizeScale = (1.0 - desired_eye_x * 2) * desired_img_width / eye_dist
```

Running Data Collection

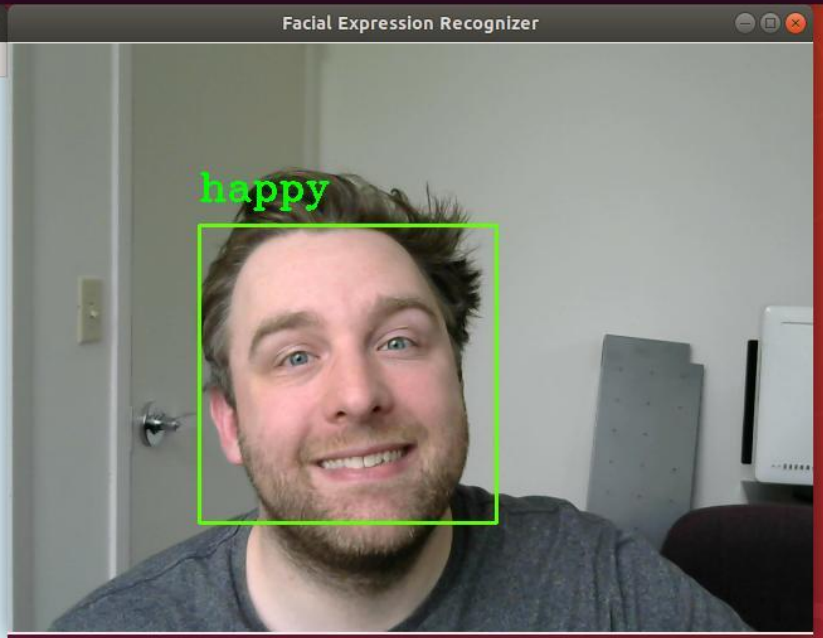


Running Training

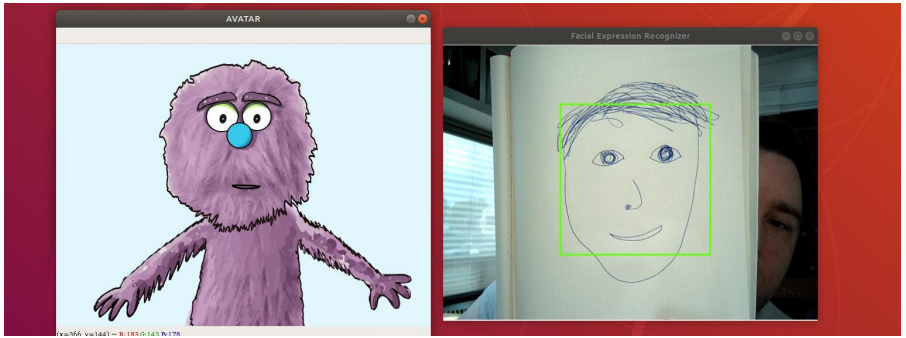


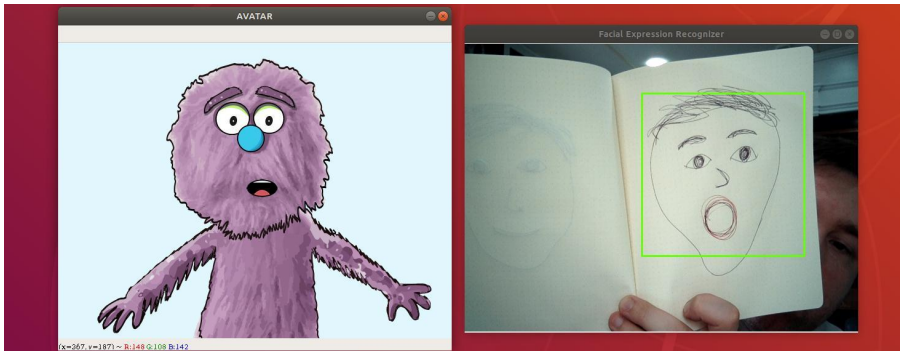
Running The Demo

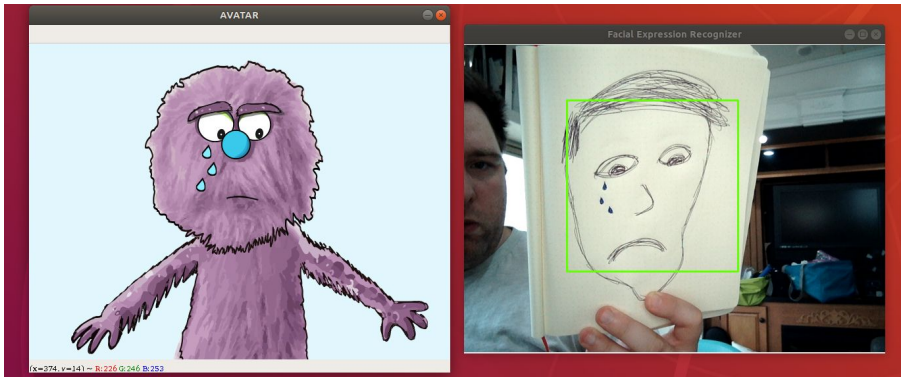
```
tyler@macbookpro:~/Documents/Github/Avatar/Avatar_emo$
tyler@macbookpro:~/Documents/Github/Avatar/Avatar_emo$ python3 avatar_emo.py demo --classifier /home/tyler/Documents/Github/Avatar/Avatar_emo/data/tyler2/
<class 'numpy.ndarray'>
(480, 640, 3)
```



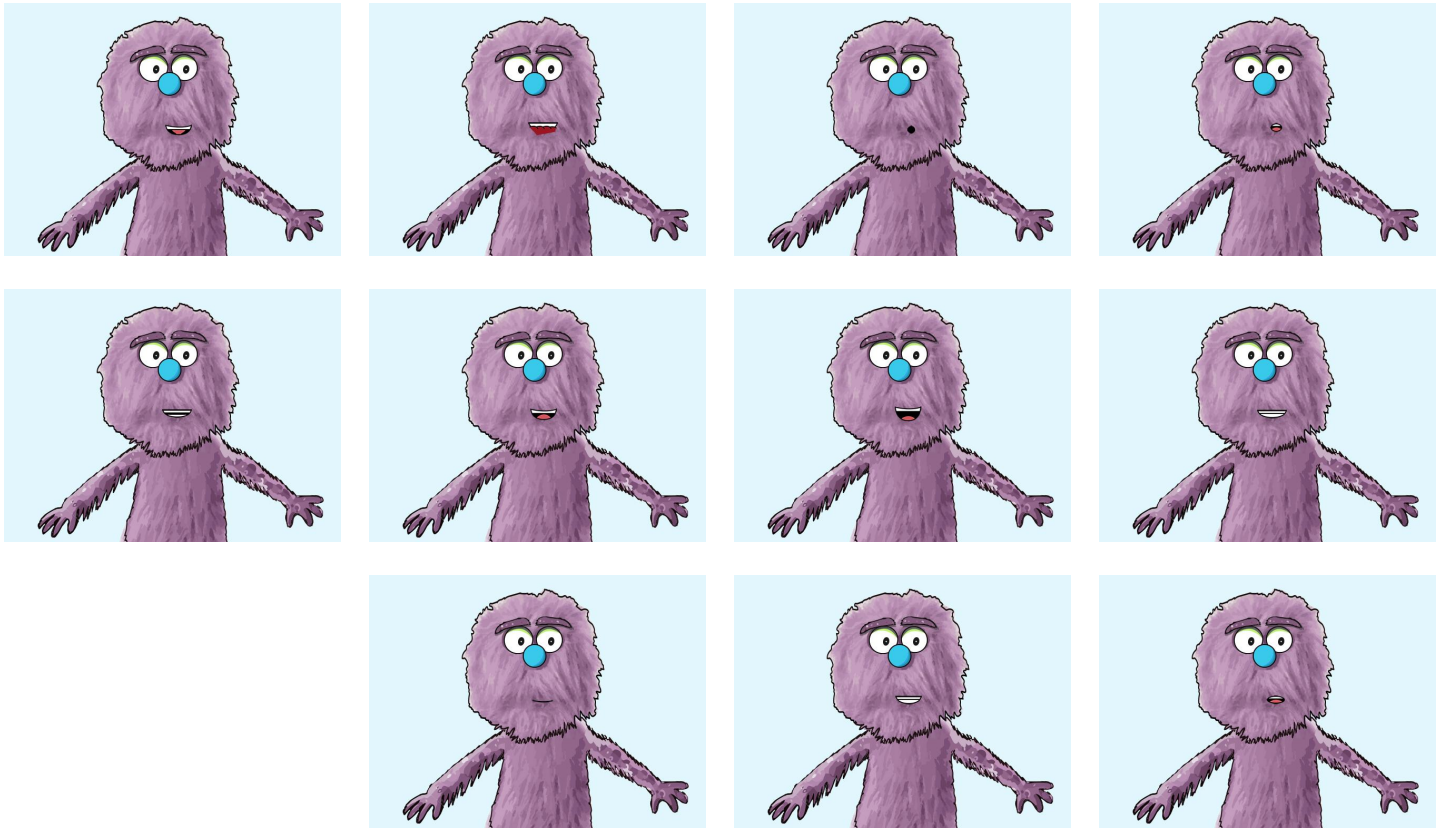
Works on Drawings as well







Reading and Displaying Talking



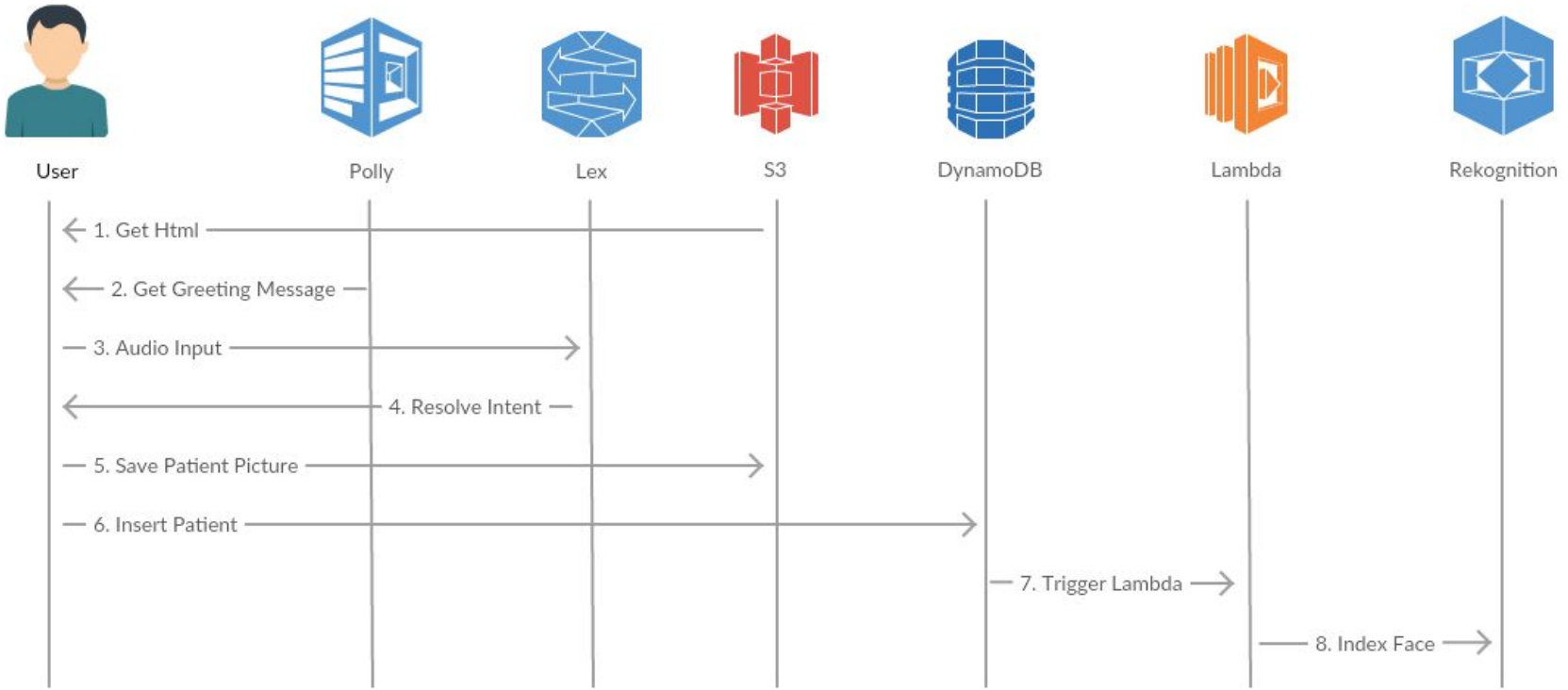
- Mouth shapes for common phonetic sounds.
- Adobe Character Animator analyzes audio to translate sounds into these shapes.
- These positions were added as options for the collection mode in our emotions program.
- When trained and running on this model, our Avatar will look like it’s saying the same things as the user.

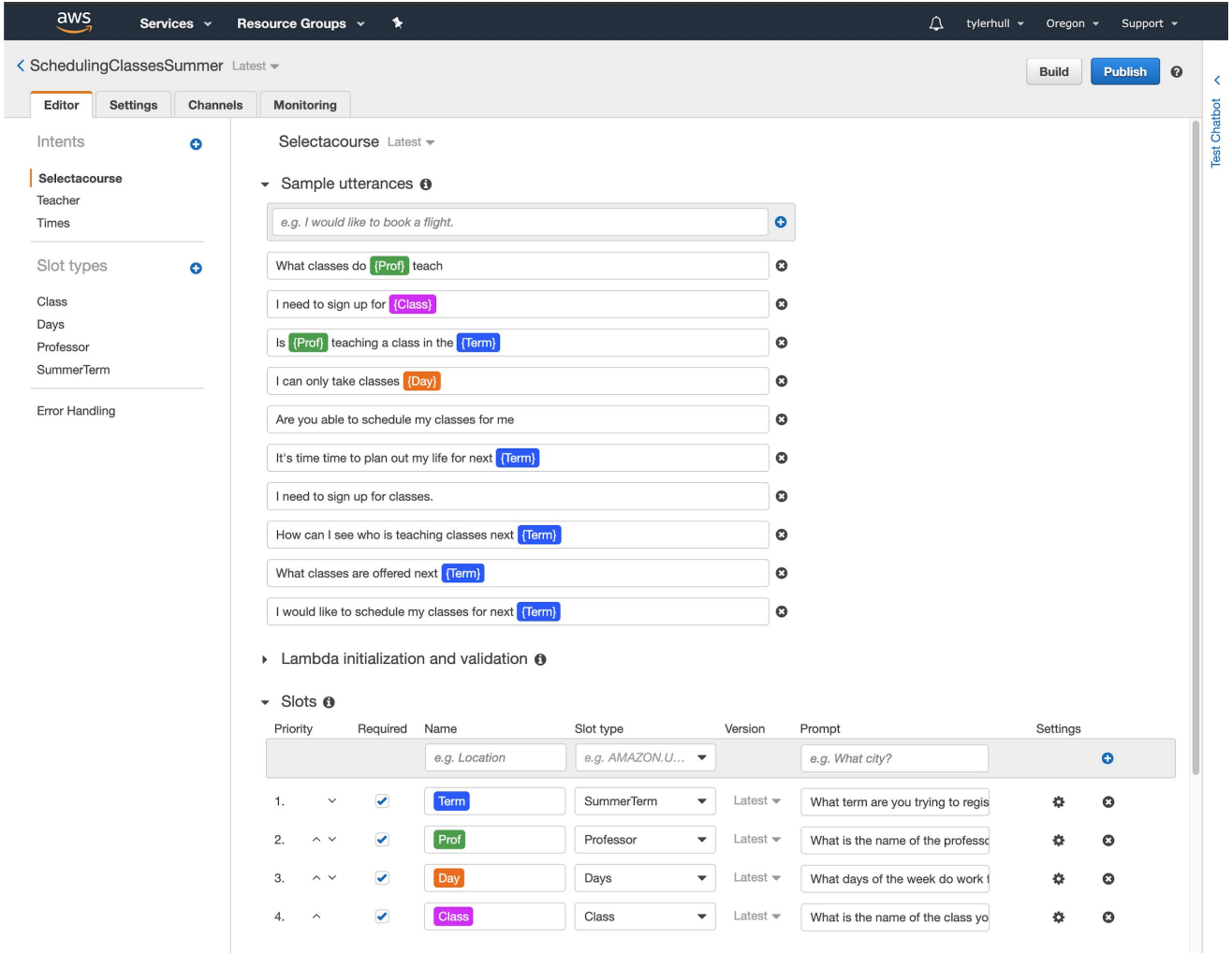
NLP and Machine Learning

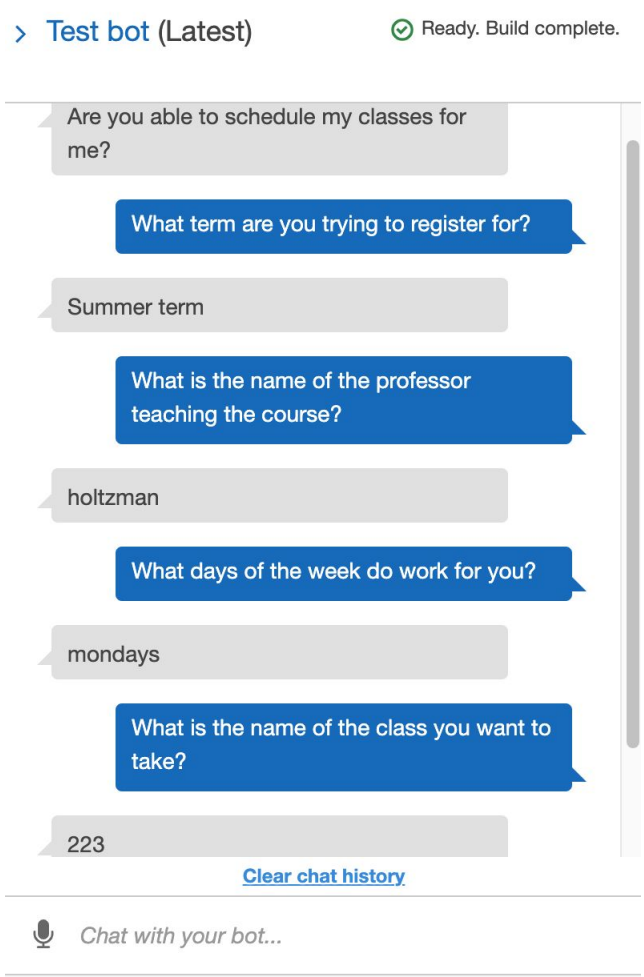
What is Amazon LEX

- An AWS service for building conversational interfaces
- Uses Amazon Polly for Automatic Speech Recognition and Natural Language Understanding.
- Built on the same learning models as Amazon Alexa.
- Integrates with other services like Call Centers, Lambda Functions, Polly, Databases, and Rekognition.

Example Program Flow




aws
Services
Resource Groups
tylerhull
Oregon
Support
SchedulingClassesSummer
Latest
Build
Publish
Editor
Settings
Channels
Monitoring
Test Chatbot
Intents
Selectacourse
Teacher
Times
Slot types
Class
Days
Professor
SummerTerm
Error Handling
Selectacourse
Latest
Sample utterances
e.g. I would like to book a flight.
What classes do {Prof} teach
I need to sign up for {Class}
Is {Prof} teaching a class in the {Term}
I can only take classes {Day}
Are you able to schedule my classes for me
It's time time to plan out my life for next {Term}
I need to sign up for classes.
How can I see who is teaching classes next {Term}
What classes are offered next {Term}
I would like to schedule my classes for next {Term}
Lambda initialization and validation
Slots
Priority
Required
Name
Slot type
Version
Prompt
Settings
e.g. Location
e.g. AMAZON.U...
e.g. What city?
1. Term SummerTerm Latest What term are you trying to regis
2. Prof Professor Latest What is the name of the professc
3. Day Days Latest What days of the week do work
4. Class Class Latest What is the name of the class yo



Inspect response

Dialog State: Fulfilled

Hide

Summary Detail

RequestID: 99dfba95-9f39-4c57-a908-24b1d9f74992


```
{
  "dialogState": "Fulfilled",
  "intentName": "Selectacourse",
  "message": "Thank you for your request. \nIs there anyt
  "messageFormat": "CustomPayload",
  "responseAttributes": null,
  "responseCard": null,
  "sentimentResponse": {
    "sentimentLabel": "NEUTRAL",
    "sentimentScore": "{Positive: 3.1203692E-4,Negative:
  },
  "sessionAttributes": {},
  "sessionId": "2020-06-14T07:38:48.920Z-mMFTYIVr",
  "slotToElicit": null,
  "slots": {
    "Class": "223",
    "Day": "mondays",
    "Prof": "holtzman",
    "Term": "term"
  }
}
```

2 Main ways to connect to your Python Program

Use the Runtime Service with Amazon's boto3 API calls

class **LexRuntimeService.Client**

A low-level client representing Amazon Lex Runtime Service:

```
import boto3

client = boto3.client('lex-runtime')
```

Can Send and Receive Audio to Polly through “inputStream”

```
response = client.post_content(
    botName='string',
    botAlias='string',
    userId='string',
    sessionAttributes={...}|[...]|123|123.4|'string'|True|None,
    requestAttributes={...}|[...]|123|123.4|'string'|True|None,
    contentType='string',
    accept='string',
    inputStream=b'bytes'|file
)
```

Use the Amazon CLI to Send and Receive Messages

```
aws lex-runtime post-text \
    --region region \
    --bot-name OrderFlowersBot \
    --bot-alias "\$LATEST" \
    --user-id UserOne \
    --input-text "i would like to order flowers"
```

```
{
    "slotToElicit": "FlowerType",
    "slots": {
        "PickupDate": null,
        "PickupTime": null,
        "FlowerType": null
    },
    "dialogState": "ElicitSlot",
    "message": "What type of flowers would you like to order?",
    "intentName": "OrderFlowers"
}
```