CSE 379: Final Project Deliverable

Team Members: Kira Gobes and Christie Altadonna
Advisor: Mooi Choo Chuah

Project Summary / Introduction & Motivation:

Children with autism spectrum disorder (ASD) often develop communication skills at different rates than children without ASD. Because of this disadvantage, it's harder for children with autism to keep up in school, make close friends, and communicate their goals. We want to help develop a solution for a fun and safe environment to assist in enabling these children to develop crucial life skills. Our project utilizes new technologies to make a difference and help children with ASD develop skills to keep them up to speed with their peers. We joined the LILI project at Lehigh University that has been worked on for several years. The LILI (Lehigh Instrument for Learning Interaction) robot has speech recognition abilities and interactive storytelling features. LILI's goal is to use interactive storytelling with autistic children to help them practice and get comfortable with interactions and communication in an engaging way. Users will be able to play through different stories, talk to the robot, and even design their own stories. As a team this semester, we were focused on creating a more developed user experience and function so that users have more interaction and thus more fun and meaningful experiences with LILI. Through a story building GUI and multiple speaker recognition, LILI's interactive potential boosts exponentially. These two features we added allow the user to create their own stories to play with LILI, as well as play with others during LILI's game play.

Project Planning:

COMPONENTS

1. User ability to create a new story and choose how many scenes they want per story.
   a. Within each scene:
      i. Choose location
      ii. Choose activities
      iii. Choose characters
2. Unexpected random events incorporated into activities
3. Ability to edit existing stories or play previously created stories
4. Option to randomize scenes or activities
5. Option to choose multiple players
   a. With at least some functionality of incorporating multiple players or into a story

TECHNICAL REQUIREMENTS

Development Environment: Ubuntu
Programming languages: Python, Java

Frameworks and Libraries: PyQt4, SciKit Machine Learning
Performance Requirements: Functional GUI with several preloaded options and 70% accuracy for multiple player speech recognition
Testing methods: Unit tests, integrated testing

Software Design:

DEVELOPMENT ENVIRONMENT

The application was developed in a virtual machine with an Ubuntu OS using the Linux terminal. It also utilizes external libraries and frameworks such as PyQt for the Python GUI and SciKit Machine Learning.

HIGH LEVEL DESIGN

The design we added to LILI has two major components, the Graphical User Interface and Speech Recognition.

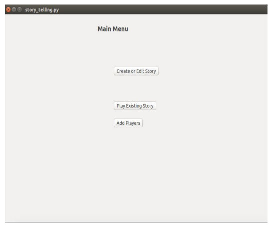
Graphical User Interface: This is built using Python and the Python toolkit PyQt4. It allows users to create their own stories or edit an existing one, as opposed to being created by the developer in a code file. The goal was to create a user interface that is simple to use and allows users to develop entertaining stories.

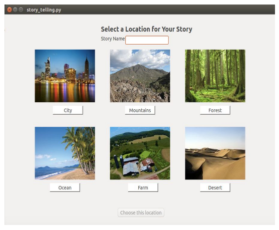
Speaker Recognition: Speaker recognition mode will be activated from the start screen. Users will have the option to "Add Players." First, users will input how many players they would like in their game. LILI will iterate through each player and first ask them for their name to give them an identifier. We then ask the player to say a couple of sentences that will help us identify them. This recording will allow the machine learning methods to identify specific characteristics in the player's voice to categorize them as one unique player.

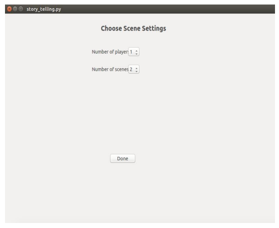
To accomplish this, we identified an external GitHub library created by Yuxin Wu that we implemented. The Voice Activity Detection that it uses is Long-term Spectral Divergence. This method is an approach to handling background noise coming in at the same time as a voice by comparing longer-term spectral envelope with the noise spectrum. For a model, it utilizes a Gaussian Mixture Model (GMM). GMM's are commonly used to model people's acoustic features. This library uses a Universal Background Model (UBM) that is a GMM but trained on a huge data set that will be able to recognize distinctive features. We implemented this library into LILI so that voices can be identified and then saved.

These steps will be repeated for each player. After these steps are complete, each of the new players will be saved in the system. Now, when users create a new story, they can select how many players they want. LILI will then list all of the recognized player options for them to choose from for that story.

Implementation Results:

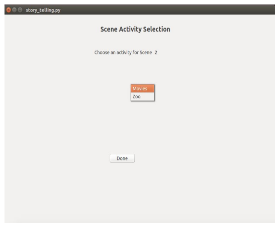






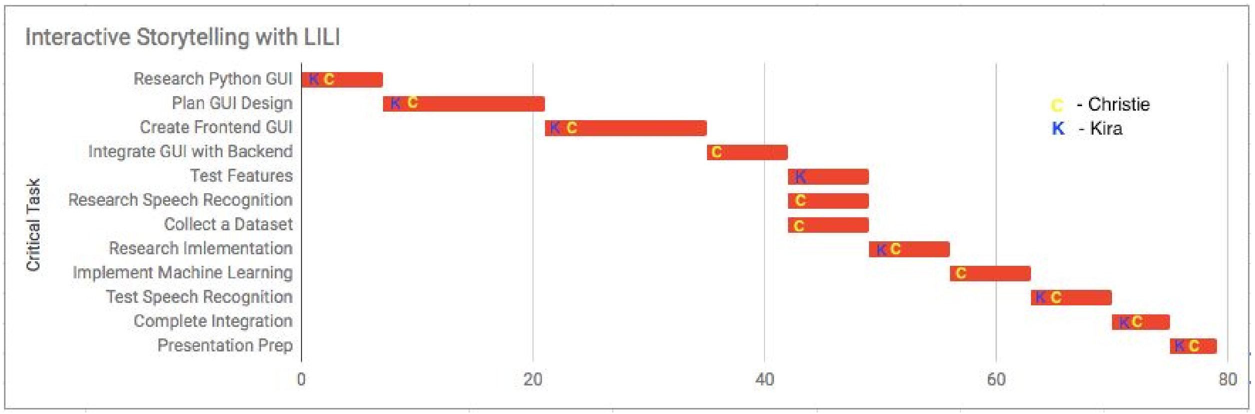
Number of player: 1

Number of scenes: 2



Choose an activity for Scene 2

Gantt Chart:



Verification & Validation:

Testing Early and Often: One of the biggest challenges for this project was working with many existing classes that previous programmers had created. This made it essential to be cognisant of testing each part of each component that we added. Because we were not totally familiar with all aspects of the existing code base, sometimes testing involved sorting through a number of classes to figure out if issues were a result of our new additions or previous pieces. For example, when creating the GUI, we carefully tested each screen before moving on to the next. This helped us catch errors in our methodology that would have been more challenging to fix later on, such as a way to iterate through showing the scene activity selection screen for each number of scenes the user wants.

Testing Outside System: In other cases of components, it was more efficient to create simulations of the piece we wanted to add outside of the LILI code base, and then add it in once we were confident in it’s effectiveness. In particular, the speaker recognition feature, we spent much of the time developing it outside of the whole code, because there were so many pieces of it that we needed to test. If we had done it with the whole story, we would have to run the whole story every time we wanted to test it. The result of this worked well for the most part. Enrolling a speaker was easy to implement in once we created it; but the predict feature for recognition worked differently once we added it in. Because we were very comfortable in our own creation of it, debugging it inside the whole system was more manageable.

Future Work:

As work on this project continues, there are several components that we think should be added and expanded on.

- Edit existing story: users should be able to take a story that was previously created and modify it with ease
- User testing: the system and user interface for the GUI should be tested with autistic children and be adjusted as needed
- Faster image loading: we've experienced some issues with our images and GIFs loading slower than we would like. We suggest future programmers to look into wifi connectivity here and if there is a connection to slower Lehigh wifi times to slow game loading.
- Multiplayer gameplay integration: Our system allows new speakers (players) to be added to the system and then recognized during the story gameplay, but the next step will be to have the specific user speaking influence the storyline in a specific or have different storylines interact with each other
- Add more scenes and locations to the story: currently there are only three scenes which are associated with the one location - City. (park, movies, zoo)

GitHub Link to our Application Source Code:

https://github.com/kgobes/SeniorCapstone

Link to Youtube Demo of our Application:

https://www.youtube.com/watch?v=noWATbS0h4I&feature=youtu.be

Documentation:

https://github.com/kgobes/SeniorCapstone/blob/master/ReadMe.pdf

(PDF version and .md file on our Github)

Weekly Updates Website:

https://sites.google.com/lehigh.edu/interactivestorytellingproject/