Software Documentation

Introduction

Amicus is a software prototype of BMW's in-vehicle Intelligent Personal Assistant (IPA). The software was developed primarily for experimental, demo and user-study purposes. Amicus was developed by a group of five students at Stanford University, as an attempt to experiment with various interfaces where dynamic visual rendering of the IPA is made possible. Rather than simply loading up an visual model of the IPA and applying a preprocessed and avatar-specific animation the model, this software has experimented with a programmatic approach where the program maps any visual model to perform its baseline animations.

The main benefits of this programmatic approach include:

- Highly scalable animation design process
- Introduce a wide array of customizable animation designs on many different visual IPA models

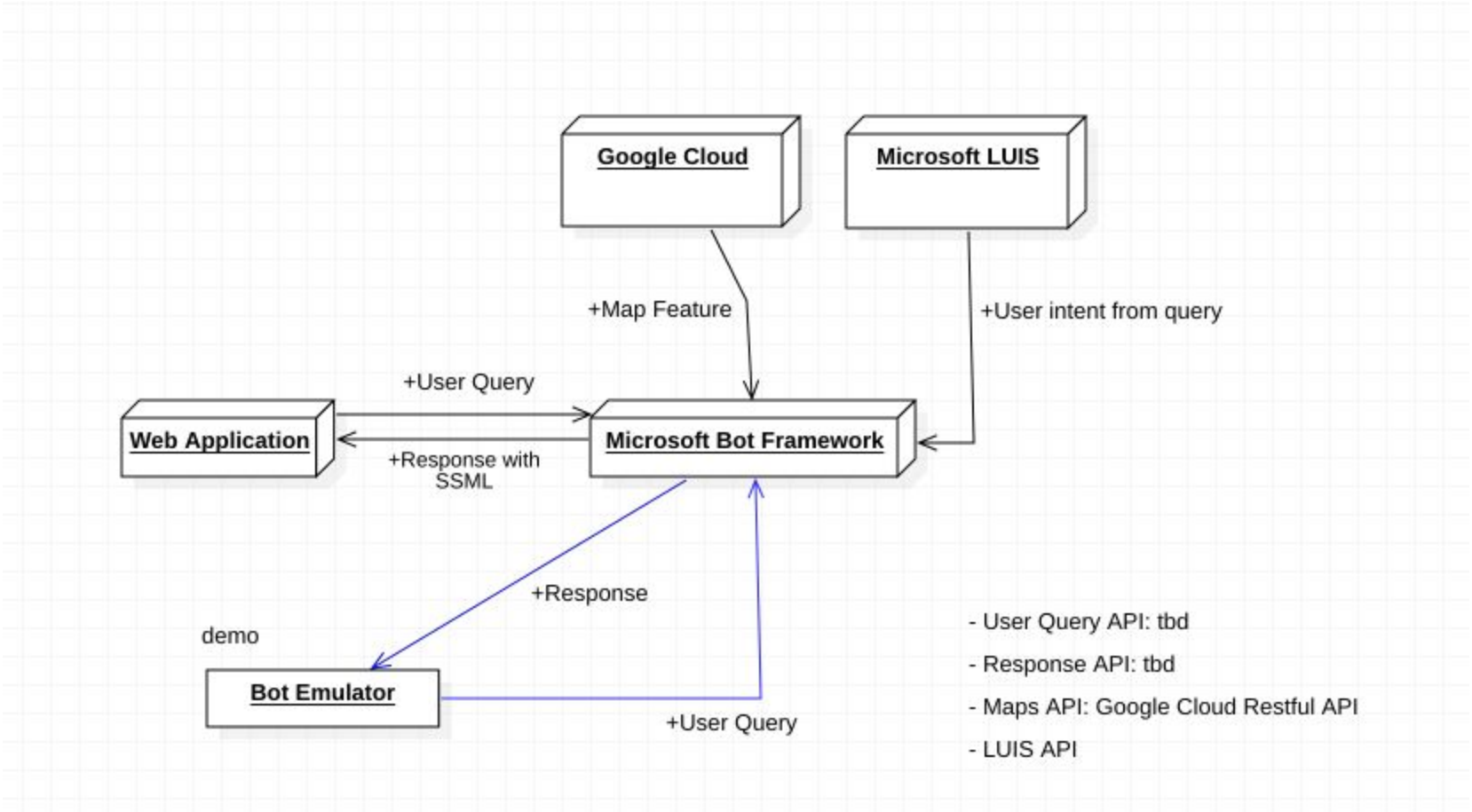
By speeding up the process of animation design, the range of animation that this IPA can perform will increase significantly. Through a wider range of animation available, the visual likeability of the IPA is expected to increase in some proportion.

Functionality

The following will detail the general capabilities of Amicus:

Frameworks

Amicus has adopted a number of framework in building each module. Careful considerations were made in designing the overall architecture. However, as point of disclaimer, the architecture is optimized for fast iterative development and is not fit for scalable production. Since Amicus was simply a experimental prototype, rapid prototyping was prioritized over stability. Developing for production would require additional stack layers above the current framework. Amicus has employed the following software framework:



Microsoft Bot Framework

Amicus's bot is based on Microsoft Bot Framework and is currently implemented so that the instance can run on several local server. Multiple instances of the bot can run on multiple local machines and all those instances are synchronized through the PubNub API. This PubNub API allows the Microsoft Bot Emulator to receive the text of the user's input and allows the Microsoft Bot Framework to send the bot's response to the web interface.

From the Model-View-Controller (MVC) architectural pattern, the Microsoft Bot Framework serves as the Model and the Controller role. Amicus has adopted Conversation dialog design and logic handling format from the aforementioned framework. Likewise, Amicus is currently equipped with basic user intent recognition capabilities, implemented using Microsoft Azure's Language Understanding (LUIS). As training for discerning user's intent was not the focus of the development, its LUIS capabilities are bare and will need further training to expand its conversational handling. Yet, by using Microsoft Bot Framework, Amicus is opened to a range of Azure's Cognitive Services.

Microsoft Bot Emulator

The primary role of the emulator is to provide a web interface for user input, package the user's input into a restful API recognizable by the microsoft bot framework, and send & receive the APIs. Therefore, the emulator serves as a mediator that serves as a UI interface that interacts with the backend controller, developed using the Microsoft Bot Framework.

For prototype purposes, the emulator provided a quick, easy and reliable solution to connecting a frontend with Amicus's backend. Yet, the emulator is replaceable with a complete UI frontend interface.

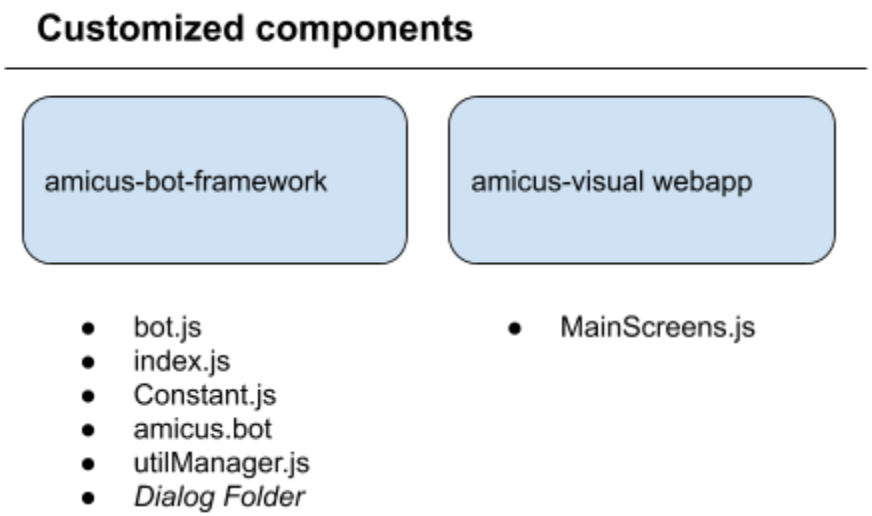
React JavaScript Web Application

While the emulator can only recognize the user's input and deliver the bot's output in text, Amicus's UI interface is equipped with speech-to-text and text-to-speech capabilities, which allows Amicus to handle verbal conversations with the users. Though the accuracy of the speech-to-text transcription process needs improvement, the web app framework provided a generic method of rendering our avatar, interact with our user verbally, and test our prototype using various emotional states of the bot. The webapp can be hosted on any chromium web browser.

Three.js

Three.js is a javascript 3D library that renders visual scene on a javascript based web interface. It is javascript native and thus, is compatible with Amicus' web application. The development of the avatar rendering and its related animations will be further advanced and experimented using this library.

Modules & Folders



Index.js: this module initializes the bot on the local endpoint at http://localhost:3978/api/messages, imports the necessary bot framework configuration, and runs "bot.js" where all the conversation events and logics are handled.

bot.js: handles the main conversation flow and evokes the necessary dialog that matches the user's intent. The call to LUIS is made from this file (but for further modularization, separating this function into the utilManager file is recommended)

utilManager.js: intended to organize any util functions used throughout the software, which includes external API calls to google cloud, google maps and Microsoft LUIS interface.

Dialog Folder: a set of dialog classes where each deals with a particular conversation flow. Each class is implemented using waterfall dialog. These dialogs are only built for demo purposes are not suited for production.

MainScreen.js: This file handles the visual portion of the amicus bot. It renders the main web application screen of the bot. The web application can be hosted locally.

Installation and Dependencies

1. Open cmd
2. Git clone the repository from this URL: https://github.com/cs210/BMW-2
3. Navigate to $ROOT/Software-Development/amicus-bot-framework/ and run the following command:

$npm install

4. Navigate to $root/Software-Development/amicus-visual-webapp/ and run the previous unix command (2)
5. Create or activate your selected conda environment
6. Navigate to $ROOT/Software-Development/ and run the following command:

$pip install -r requirements.txt

7. To start the web app: navigate to the amicus-visual-webapp/ folder and run:

$npm start

8. To start the microsoft bot framework: navigate to the amicus-bot-framework/ folder and run the previous unix command (7)