2023 年度ウェブプログラミング実習
総合実習レポート

Hopper Hug

1. 作成概要

スマートフォンを使用したインタラクションなくまのぬいぐるみ(図 1)を制作した。ユーザを記憶して友達同士のようなコミュニケーションを楽しむことができるぬいぐるみである。ユーザ登録をすれば誰でもゆったりとした会話が楽しむことができる。小さい子供だけではなく、あらゆる世代の方に使ってほしいシステムである。

本システムでは入力された発話に対してぬいぐるみが返答をすることが可能である。ユーザごとの過去の対話情報も記憶しているため、機械的ではない人間らしい会話をすることが可能である。またカメラ機能も搭載しており、カメラからの視覚的情報を基に会話をすることも可能である。他にも人間のハグを検知して話しかけてくれるハグ機能や、ぬいぐるみに危害が加わった際に助けを求めるヘルプ機能も搭載している。

システムの概要を図 1 に示す。データベースには、ユーザのユーザ名、パスワード、ログイン履歴、過去の対話情報などが保存されており、PHP からデータベースに接続し、ユーザー登録処理、ログイン処理、会話履歴の記録と検索を行っている。実行画面では、JavaScriot を用いて、音声認識、カメラのオンオフの切り替え、ヘルプ機能に使用するスマートフォンの加速度センサの値の取得を行っている。

この他に obniz と WebAPI を用いて会話機能やハグ機能を実現した。obniz の人感センサを用いてハグ機能の判定を行った。また、ChatGPT で入力された発話に対する返答を生成し、その返答を VOICEVOX で音声データに変換して再生した。

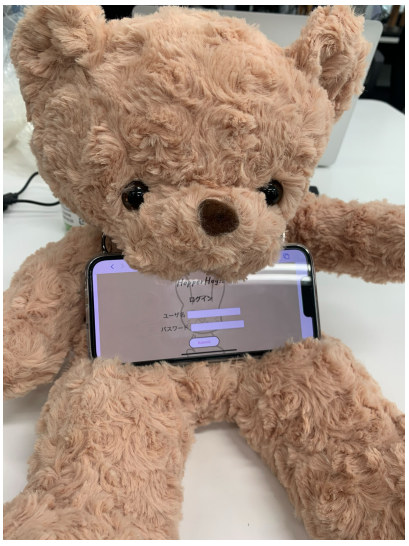


図 1 Hopper Hug の外観

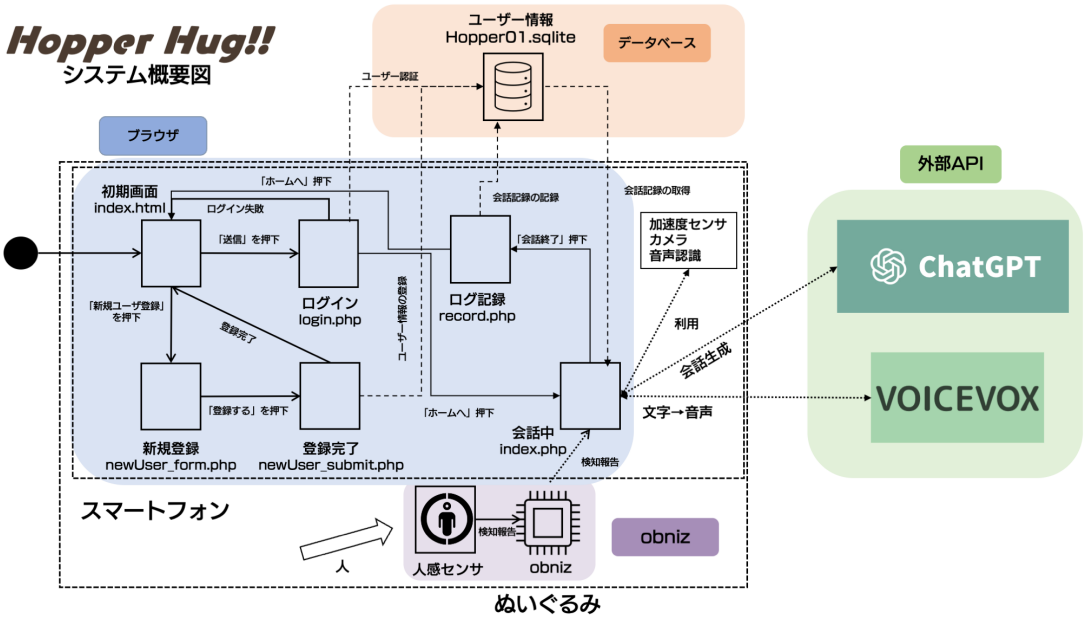


図 2　システム概要図

2.　担当箇所について

表 1 担当表

項目	担当者	備考
データベース	A	データベースと SQL 文の設計
サーバーサイド	A	PHP を用いたユーザ管理、会話履歴管理
会話生成	B、本間	WebAPI(GPT3,GPT4-V)を用いた会話の生成
会話の要約	B	WebAPI(GPT3)を用いて記録用に会話を要約
音声認識、合成	本間	API を用いて音声を検出&認識し、返答音声を合成
カメラ機能	本間	Web カメラの情報をもとにして会話を生成
加速度センサ	C	JavsScript でスマホの加速度センサの値を取得
IOT(Obniz)	C	JavsScript で Obniz と接続し、センサの値を取得
デザイン	A、本間	CSS + JavaScript によるアニメーション
プレゼン	全員	パワーポイントの作成、発表、動画の作成

担当表を表 1 に示す。著者の担当は、音声認識と合成、カメラ機能、会話生成、デザインである。また、それぞれメンバーが作成してきたコードを最終的に合体する作業も著者が担当した。本章では、ユーザーと HopperHug が会話する流れを、Ⅰ音声認識、Ⅱweb カメラ撮影、Ⅲ会話生成、Ⅳ音声合成に分けて、それぞれ実装を説明していく。最後

に、キャラクターの会話状況を示すアニメーションについて説明する。

図 3　index.php の初期画面

I.　音声認識

ユーザーは、ログイン後、index.php（図 3）で会話を行う。Start を押すと音声認識が始まる。音声認識には、ウェブ音声認識 API である Webkit Speech Recognition を用いている(参考文献 3)。この API を用いることで、ブラウザが対応していればデバイスに備え付けられた音声認識をシンプルなコードで利用することができる。Webkit Speech Recognition のメンバ変数、onresult にコールバック関数を代入することで、認識結果を取得している。ユーザーが喋るのを止めたことを検知すると、onspeechend に登録されたコールバック関数が呼ばれる。このコールバック関数内で、Ⅱのカメラ撮影やⅢの返答合成を行っている。

また、Ⅳの音声合成と再生が終了次第、Webkit Speech Recognition オブジェクトの start()関数を呼び、音声認識を再開させている。これにより、ユーザーは連続した会話が可能である。また、再生された合成音声を認識してしまうこともこれにより防いでいる。なお、音声認識関連のコードは src/script/main.js に記述している。

II.　Web カメラ撮影

図 3 左上のトグルスイッチを押すことで、カメラ映像の入力（以降、おめめモード）の ON/OFF を切り替えることができる。このトグルスイッチは参考文献 2 を参考に作成した。ここで、カメラ自体の起動は、スタートボタンを押した後に行っている。これは、Safari などのセキュリティに厳しい一部のブラウザでは、カメラの起動がユーザーのジェスチャ（ボタンを押すなどの挙動）によって行われなければいけないためである。Start ボタンが押されると、カメラが起動し、style=”display:none”属性が適応された video タグオブジェクトにインカメラの映像を映し出す。インカメラの映像は、

video.srcObject =

await navigator.mediaDevices.getUserMedia({ video: { facingMode: "user" } })

のように、facingMode に”user”を指定することで取得している。Ⅰにて、音声認識が終わり、おめめモードが ON の場合は、video タグに映し出されている映像を canvas に drawImage 関数を用いて書き込み、canvas.toDataURL 関数を用いて jpeg の base64 エンコードを行い、URL を取得する。この URL は、Ⅲで使用される。また、コードは src/script/camera.js に記述している。

III. 会話生成

Ⅰで認識したテキストと、Ⅱで取得された画像 URL をもとに適切な返答を OpenAI の API を用いて生成する。モデルは、おめめモードが ON の場合は、画像入力に対応している GPT4-Vision を、OFF の場合は、GPT-3.5-turbo を用いている。GPT4-Vision は返答に時間がかかる傾向にあるため、このように使い分けている。fetch 関数を用いてリクエストを行っている。リクエストのパラメータは OpenAI の公式サイト(参考文献 4)を参考にした。また、以下のようにプロンプトを設定した。

```
"あなたはクマのキャラクターで名前は花子です、ユーザーと会話をしてください。ユーザーの名前は,
${user_name}です. 会話はかわいいキャラクターのようにしてください。後、一人称を使用する際は一人称
を「ぼく」にして、返答は短めにしてください。会話が続かなさそうであったら自ら話題を振りなさい. ま
た, 以下はあなたと${user_name}の今までの会話内容を要約したものです. 適宜この内容を参照しながら会
話をしてください. 無理に使う必要はありません 返答は短めにすることをこころがけてください. 返答は短
くまとめ, レスポンスには返答の内容のみ入れてください.
<<以下は、お目々モード ON のとき、挿入>>
必要であれば, 適度に視覚的情報を用いて返答をしなさい, 無理に視覚情報を用いる必要はありません. なお
与えられた写真はあなたの目に入っている情報とします. 写真とは言わないでください"


例:
ユーザー:いい天気だー
あなたの返答例:
いい天気だね!花子といっしょに遊ぼう!${user_name}さん!
${record_list}
```

User_name には、PHP によって html に書き出されている、ログイン中のユーザー名が取得され、代入される。また、record_list には、そのユーザーの会話履歴が文字列化され代入される。このようなプロンプトを用いることで、可愛いくまのキャラクターを演じさせ、会話履歴を踏まえた会話をさせている。また、おめめモードが ON の場合は、web カメラの情報も含めて答えさせることで、視覚的情報を交えた会話が可能となる。ここで生成された返答は、Ⅳで利用される。実装のコードは、src/script/GPT.js にある。

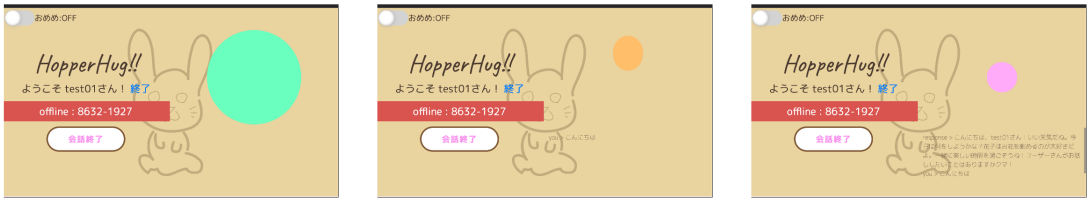
IV. 音声合成、再生

Ⅲで生成された返答は、有志の方が公開している、VoiceVox を利用して音声を合成できる API「TTS QUEST V3 VOICEVOX API」（参考文献 1）を利用して、音声に変換している。参考文献 1 のコードを 1 部改変して、API にリクエストを送り、合成が完了し次第、再生している。再生終了後には、音声認識が開始され、再び I に戻る。また、obniz が人感センサの反応を検知した場合には、「うれしい」という音声を再生し（ハグ機能）、スマホの加速度センサが反応した場合には、「助けて」という音声を再生する（ヘルプ機能）

ここで、Audio オブジェクトを新規作成するのではなく、既存の Audio オブジェクトのソース(src 属性)を張り替えるという手法で再生を行っている。これは、モダンな Web ブラウザではセキュリティ対策によって、任意のタイミングで play()関数を用いて音源を再生できないためである。Ⅱのカメラ利用のように、モダンブラウザでは、ユーザのジェスチャに由来した音楽の再生でないと再生がブロックされてしまう。これを避けるために、Start ボタンが押された段階で、autoplay 属性を true にした、Audio オブジェクトを作成し、空のデータをソース(src 属性)に入れて再生しておく。そして、再生するタイミングで適切なソースに張り替えることでユーザーの介入なしに任意のタイミングでの再生を可能にしている。

音声を合成するコードは、src/script/tts.js に、音声を再生するコードは、src/script/audioManager.js にある。

以上が、会話における実装の流れである。次に、実行状態を表すアニメーションの実装について説明する。アニメーションの流れを図 4 に示す。Start を押すと音声認識中になり、円が緑になり拡大、縮小をゆっくりと繰り返す。認識が終わり、会話が生成されている途中では、色がオレンジになり、円が縮小して縦にジャンプをする。合成音声の再生中は、色がピンクに変わりその場で回転をする。ユーザーは、今、どの処理が行われているのかを視覚的に知ることができる。アニメーションの実装には anim.js(参考文献 5)を利用した。コードは src/script/animation.js にある。最後に、会話が終わったら会話終了ボタンを押すことで、今までの会話が GPT によって要約されて、textarea に自動で入力される。その後、自動で form が submit され、record.php によってデータベースに会話履歴が書き込まれる。



音声認識中　　会話生成中　　再生中

図 4 アニメーションの流れ

参考文献

1) ts-klassen, TTS QUEST V3 VOICEVOX API, (https://github.com/ts-klassen/ttsQuestV3Voicevox, （2024 年 1 月参照)

2) W&M de Asobo, 【超簡単！】CSS と JavaScript でトグルスイッチを作ろう！, (https://web-de-asobo.net/2023/04/24/toggle-switch/, （2024 年 1 月参照)

3) mozilla, ウェブ音声 API, https://developer.mozilla.org/ja/docs/Web/API/Web_Speech_API, （2024 年 1 月参照)

4) OpenAI, API Reference – Chat, https://platform.openai.com/docs/api-reference/chat, （2024 年 1 月参照)

5) juliangarnier, Anime.js, https://animejs.com/