24 POINTS

XIUMEI XUE

2024.03.20

01 What does it do?

- Rule Introduction
- Rundown Brief

Rule Introduction

- You will be given 4 random cards from 1 to 9 (“A” refers to 1), and three entry levels vary
- You can use + − x / to generate a result of 24, and each card should only be used once
- Decimal, fraction or negative number can appear during operations
- Also, a timer will be set according to your chosen level (Easy: 3min | Medium: 1min30s | Hard: 60s)
- If you can get 24 in limited time, you win, or you fail.

Dialogue System
+
Page Interaction
=
Dialogue-Driven Interaction

Rundown Brief

YOUR RESULT:

display your expression/hint

OPERATOR SELECTION: + − × / ( )

TOP SETTINGS: Clean Confirm Change Hint

02:38 countdown

02 Technicalities

- Platform and Technologies
- Library

Technicalities

Platform

- Web Platform based

Backend Technology

- Node.js (LTS version)
- Package Manager: Yarn
- Azure service for ASR&TTS&NLU

Frontend Technologies

- JavaScript (Xstate, vue.js framework)

  azure.js – export relevant KEY
  dm.js – manage dialogue–driven interaction
  main.js – control the main logic and flow of the game

- HTML

  index.html – the main entry point of the application, containing the structure and layout of the web page

- CSS

  style.css – define the styling rules for HTML elements

Technicalities| Library

- Import a third-party JS library:
  Poker.JS
  https://tairraos.cloud4v.org/Poker.JS/#english-version-readme

Usage

First, load poker.js:

```
<script src="poker.min.js"></script>
```

Then there have 3 ways to create card by your choice

Way 3, Draw card in your own canvas

Add your own canvas to DOM

```
<canvas id="myowncanvas" width="1280" height="720"></canvas>
```

Get canvas 2d object and draw card

```
var canvas = document.getElementById('myowncanvas').getContext('2d');
canvas.drawPokerCard(10, 10, 120, 'hearts', '6');
```

03 Recap & Outlook

- Challenges
- Relation to course contents
- Future Work

Recap | Challenges

Variable sharing

- variable assignment within dmMachine => no change outside

```
dmActor.subscribe((state) => {
    targetTime = state.context.targetTime;
}); // always reflects the latest state of the targetTime value in the dmActor
```

Utterance Collapse

- Use after for delayed transitions– "Have you thought of a solution?"
- Player action *unpredictable*

Interaction Triggered Utterance(Solved)

```
function successUtter() {
    dmActor.send({
        type: "WELLDONE"
    });
}
```

Course contents

Most useful part(s)

- Understanding and Implementation: ASR+TTS+NLU+DM
- Lightweight Product Development Capability

Statecharts as an implementation framework

- Reduced state counts–guard transition, history…
- Structural
- Rule–based– more flexibly?
- Understandable

*(Xstate) Gammar Acquisition!

Development process

- Design the main logic => Include the dialog flow => Combination and coordination
- Test by myself, by peers => Bug fixing

*Not relates to ethical concerns

Future work

- In general, my game is almost end-to-end completed
  - Improvements on level distinction
  - Enhancing the coordination between Dialogue and Interaction
  - Exit & Replay

- About conversational features
  - Assign different voice styles in different occasions – e.g. succeed => “cheerful”, fail => “sad”
  - Fix/avoid utterance collapse as much as possible – adjust delayed time flexibly?
  - Mixed-initiative – collect demand by input rather than selection in given choices

T H A N K S