Word game Ghost for the robot Pepper

Samuel Gazda, 469083
Contents

• Robot Pepper
• Word game Ghost
• Choregraphe
• Implementation of the game
• Possible improvements of the implementation
Robot Pepper

- Social humanoid robot by softbank robotics
- Provides preception modules for communication with surroundings (including 2D/3D cameras, touch sensors, speech recognition)
- Operates on open and fully programmable platform
Ghost

• Ghost is a word game for two or more players
• Each player begins with agreed number of lives (usually 5)
• Players take turn to add a letter to an existing word fragment avoiding creating a valid word
• Player loses a life if he is unable to add a letter, adds a letter which leads to non existent word or creates a valid word by adding a letter
• Players with zero life are Ghosts, and cannot participate any longer. The last player with any lives wins the game.

• Player may instead of choosing a letter challenge pick of the previous player. If the challenged player cannot produce such a word, he loses a life. Otherwise the challenging player loses a life.

• Challenging a choice ends a round and the new word is started.
• If the whole vocabulary is known, it is a game with perfect information

• With different vocabularies exist different perfect strategies

• (For example one by Randall Munroe)

https://xkcd.com/1002/
Choregraphe

• Multi platform desktop application from Softbank Robotics
• Tool for creating animations, behaviour and dialogs for Pepper
• Test this solutions on simulated robots (more or less)
• User friendly drag and drop interface but supports python code
• All the boxes are running at the same time, waiting for the impulse to begin certain behaviour (sometimes hard to debug)
• Provides easy to use tools for TTS and speech input

class MyClass(GeneratedClass):
    def __init__(self):
        """
        This part initialises main game loop, live counter and vocab for robot. Vocab is loaded from original_word_list.txt.
        """
        GeneratedClass.__init__(self)
        self.tts = ALProxy('ALTextToSpeech')

    def debug_say(self, word):
        self.tts.say(word)

Text to speech usage example
Implementation

• Available at: https://gitlab.fi.muni.cz/xgazda/pepper_ghost_game
• Combination of python script and build in box libraries
• Only Human versus Pepper version
• Pepper searches through his available words from his inner dictionary, plays random letter which doesn't make him lose immediately
• Pepper uses same vocabulary for checking for word validity and his own turn
  – Proves hard to beat
Transcript of the game with one life

Robot: Chceš si zahrát hru Duch?
Human: ano (100%)
Robot: Mám ti vysvětlit pravidlá hry?
Human: ne (100%)
Robot: Ok, můžeme začít.
Human: t (100%)
Robot: r
Robot: Tvůj tah!
Human: p (100%)
Robot: e
Robot: Tvůj tah!
Human: l (100%)
Robot: i
Robot: Tvůj tah!
Human: v (100%)
Robot: e
Robot: Tvůj tah!
Human: h (100%)
Robot: o
Robot: Prohrál sem!
Robot: Nemám žádné životy! Prohrál jsem.
Robot: Chceš si ještě zahrát?
Human: ne (100%)
Tablet Interface

- Robot provides a tablet which can show webpages
- Script creates a simple webpage with current game state (simple HTML + CSS)
- Choregraphe cannot simulate tablet (cannot check actual results)
Possible Improvements

• Modifying game so that it can be played by more players
• Restrict robots vocabulary, It is not enjoyable to always lose
• Better design of internal webpage
• Robot takes too long to sort through frequent prefixes
Thank you for your attention
Sources

https://www.softbankrobotics.com/emea/en/pepper
https://commons.wikimedia.org/wiki/File:Pepper_the_Robot.jpg
https://en.wikipedia.org/wiki/Ghost_(game)
https://images.app.goo.gl/sVjmpVe8V21TEhfj9
https://images.app.goo.gl/M78i52g2p2WtiNBJ9