**VIVEKANAND EDUCATION SOCIETY’S INSTITUTE OF TECHNOLOGY**

**Department of Computer Engineering**

****

Project Report on

**Natura.IO**

In fulfilment of the Fourth Year, Bachelor of Engineering (B.E.) Degree in Computer Engineering at the University of Mumbai Academic Year 2019-20

**Submitted by**

Faizanshah Ansari (D17C - 05)

**Under the guidance of**

Mrs. Priya R.L.

**INDEX**

**CHAPTERS**

|  |  |  |
| --- | --- | --- |
| **Sr No.** | **Title** | **Page No.** |
| 1. | Introduction | 1 |
| 1.1 Problem Definition | 1 |
| 1.2 Scope of Project | 1 |
| 1.3 User and their requirements | 1 |
| 1.4 Technology to be used | 1 |
| 2. | Literature Survey | 2 |
| 3. | Conceptual System Design | 18 |
| 3.1 Conceptual System Diagram | 18 |
| 3.2 Flowchart | 19 |
| 4. | Implementation | 20 |
| 4.1 Code | 20 |
| 4.2 Output | 23 |
| 5. | Conclusion and Future scope | 24 |
| 5.1 Conclusion | 24 |
| 5.2 Future scope | 24 |
| 6. | References | 25 |

**1. Introduction**

**1.1. Problem Definition**

It is seen many times that video game players face difficulties while trying to portray their input to the gaming system. Sometimes, it is because they are so engrossed in the game that by-mistake they give wrong input. Voice (or Written Text) is a natural interaction medium for Humans. Games capable of processing their I/O based on “Natural Human Talks” present many benefits, especially to the players in the field of Serious Games. The players that will be playing any Serious Game may or may not be an expert gamer. Thus he/she may be more comfortable while interacting using the natural human language.

**1.2 Scope of Project**

To develop tool for Natural Language I/O Interaction in Video Games.

**1.3 Users and their requirements**

* Game Developers (Gameplay Programmers): API that can be Integrated.
* Video Game Hobbyist & Professional Players: Ready-made game.
* Scientists working on interactive simulation research: Player’s behavioural via IO data.

**1.4 Technologies to be used**

**1.4.1. Tkinter:**

Tkinter is a Python binding to the Tk GUI toolkit. It is the standard Python interface to the Tk GUI toolkit, and is Python's de facto standard GUI.

As with most other modern Tk bindings, Tkinter is implemented as a Python wrapper around a complete Tcl interpreter embedded in the Python interpreter. Tkinter calls are translated into Tcl commands which are fed to this embedded interpreter, thus making it possible to mix Python and Tcl in a single application.

**1.4.2 Language Understanding (LUIS):**

Language Understanding (LUIS) is a cloud-based API service that applies custom machine-learning intelligence to a user's conversational, natural language text to predict overall meaning, and pull out relevant, detailed information.

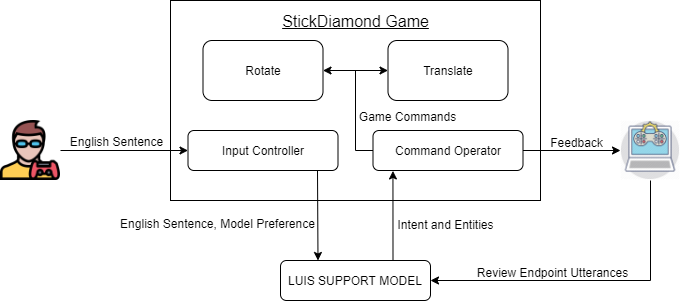
A client application for LUIS is any conversational application that communicates with a user in natural language to complete a task. Examples of client applications include social media apps, chat bots, and speech-enabled desktop applications.

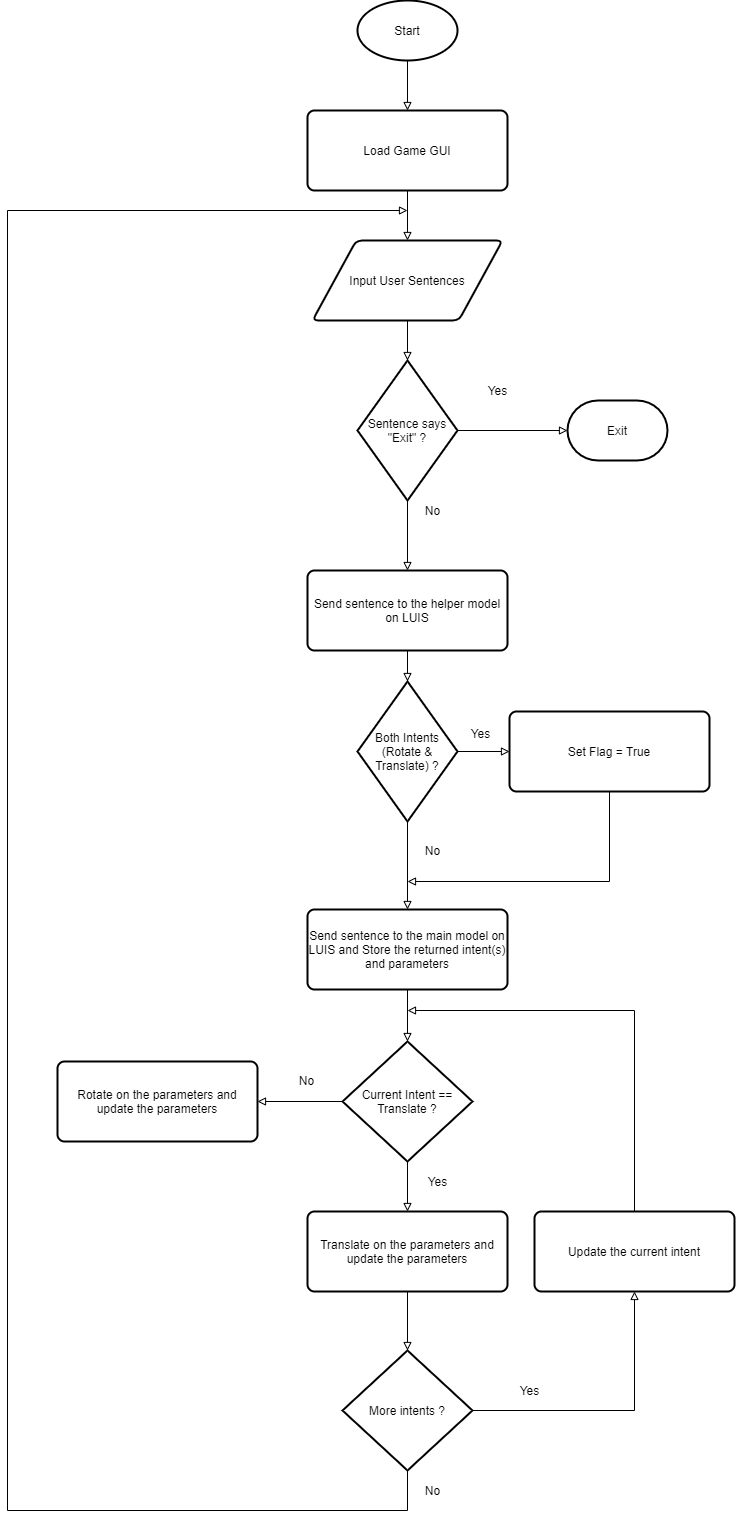
**2. Literature Survey**

|  |  |  |  |
| --- | --- | --- | --- |
| **Title** | **Overview** | **Positive Aspects** | **Limitations** |
| A Study on Natural Language Processing for Human Computer  Interaction | This paper gives an overview of eminent technological perspective and appreciation of the fundamental progress on how humans can interact with computer using Natural language.  Finally, the paper concludes with the decision on future direction for developing techniques in human computer interface and it also discusses various techniques used in each step of Natural Language Processing. | It covers all the aspect of typical NLP processing ranging from Receiving the input, Processing the Input to finally Output Generation. | It fails to provide mechanism to resolve natural language word – intent dependencies of words which are further apart. |
| UTTERANCE-LEVEL EXTRACTIVE SUMMARIZATION OF OPEN-DOMAIN  SPONTANEOUS CONVERSATIONS WITH RICH FEATURES | In this paper, they summarize spontaneous conversations with features of a  wide variety that have not been explored before.  Experiments show that the use of speech-related features  improves summarization performance. In addition, the  effectiveness of individual features is examined and  compared. | To calculate semantic similarity  between a given utterance and the conversation, the noun  portion of WordNet is used as a knowledge source, with  semantic distance between senses computed using Leacock-  Chodorow normalized path length. The noun senses were manually disambiguated  rather than automatically.  Features: MMR score, Lexicon features, Structural features, Prosodic features & Spoken-language features.  They have used ROUGE a widely used evaluation package for text  summarization to compare their results.  It show that the performance of the summarizers  improved, in general, with more features used. The use of  lexicon and structural features outperforms MMR, and the speech-related features, acoustic features and spoken language features, produce additional improvements.  While speech processing this paper focusses more on “how-it-is-said” than “what-is-said.” | If the noun senses are instead  automatically assigned to be the dominant sense, the  summarization performance is worse than tf.idf based  maximum marginal relevance (MMR) |
| Voice Recognition Scheme for Coding Console  Using Natural Language Processing-A Survey | They have developed a voice response system (VRS). It is a system interface which acknowledge to speech instructions instead of acknowledge to inputs from a mouse or a keystroke. The project offers a coding console for software programmer. Where programmer can dictate any code of C programming language and it will be written on editor along with compiler. This project is a proposal of different approach to the NLP application and replacing the traditional code typing method with smart console recognizing the programmer’s speech. | The proposed system focusses on Automatic syntax generation. Natural Language  Processing (NLP) is focusing on natural language of  coder which will automatically generate the code instead  of typing manually.  They conduct disambiguation of similar sounding words.  When the user speaks some  reserved word that is first converted into text and then this  text will be mapped with the list of structure. If any match  is found then this text is alter with reserve word, if not  then is written as it is. Now if this text is wrong and user  wants to delete that word user will speak “incorrect” or  “delete”.  A list of structure is also maintained for similar  delivery of incorrect. If spoken word is matched with that  same delivery then that word is deleted. At that time if  match is found then reserved word has its own program  construct then that Program construct generate  simultaneously. | Syntax needs to be spoken than logic.  High error–correction action rate. |
| Natural Language Processing in Serious Games: A state of the art. | The objectives of this paper are twofold: on the one  hand, providing a simple framework to enable analysis of potential uses of NLP in  Serious Games and, on the other hand, applying the NLP framework to existing  Serious Games and giving an overview of the use of NLP in pedagogical Serious  Games. In this paper they present 11 serious games exploiting NLP techniques. They  present them systematically, according to the following structure: first, they highlight  possible uses of NLP techniques in Serious Games, second, they describe the type of NLP implemented in the each specific Serious Game and, third, they provide a link to  possible purposes of use for the different actors interacting in the Serious Game. | NLP Objectives:  1) iSTART-ME:  Analyzing the self-explanation provided by the student and giving him/her a feedback represents the  critical part of this game.  2) Eveil 3D:  In this game a verbal automatic speech recognizer is trained and used to accept input from the  students.  3) I-fleg:  As it is integrated in Second Life, this game benefits from 3D graphics, virtual reality and NLP technologies provided by the environment. The test exercises are produced in a non-deterministic  way and their content is dependent on the learner’s profile and the goal to be reached.  4) BOSS (BOrder Security System):  A speech recognition  module, a speech synthesis module and an emotion recognition module are used to manage the conversation between the NPCs and the  player.  5) DeLearyous:  NLP is used in order to allow a natural language conversation between the learner and an NPC. To classify the player’s position in Leary’s Rose, the program must determine two axis of behavior: the dominant-submissive position and the opponent-cooperative position.  6) Façade:  NLP is used to improve dialogue efficiency between the player and the NPC. NLP enables pragmatic  dialogues between the player and the NPCs, with no emphasis on the syntax or the semantics of the  input sentences.  7) FearNot!:  Classifying speech acts and extracting semantic information generally represents a difficult task.  Since this game is played by children, who usually express themselves with simply sentences, the  process is simplified Consequently, the input domain will be constrained to a more restricted number  of words.  8) Mission Rehearsal Exercise:  In this game the authors ambition to create virtual NPCs with a behavior closely resembling that of  real humans. To achieve this goal they integrated a great number of AI and NLP technologies.  8 & 9) AutoMentor and Land Science:  AutoMentor replaces a human mentor by using several NLP techniques trained by human experts.  10) Operation ARIES!:  The program has been designed to use natural language conversations between the learner and  several NPCs, relying on a written form for the learner and on a written and verbal form, combined  with emotional facial expression, for the NPCs. | Games are created using old Game tools.  Very few gameplay elements are present. Most games are static (Voice/Text interaction and less object interaction). |
| Fast and easy language understanding for dialog systems with  Microsoft Language Understanding Intelligent Service (LUIS) | LUIS is entirely cloud-based: developers log into a web-site, enter a few example utterances and their labels, and then deploy a model to an HTTP endpoint. Utterances sent to the endpoint are logged and can be efficiently labeled using active learning. Visualizations help identify issues, which can be re-solved by either adding more labels or by giving hints to the machine learner in the form of features. | When a classification error surfaces in a visualization, developers have a few options for fixing it: they can add more labels; they can change a label (for example, if an utterance was mislabeled);  or they can add a feature. A feature is a dictionary of words or phrases which will be used by the machine learning algorithm. Features are particularly useful for helping the models to generalize from very few examples – for example, to help a model generalize to many types of de- vices, the developer could add a feature called ActivityWords that contains 100 words like “run”, “walk”, “jog”, “hike”, and so on. |  |
| Design and Implementation of Voice Controlled Tetris Game Based on Microsoft SDK | The framework of speaker independent voice controlled Tetris game based on speech recognition is introduced in this paper. Then the implementation of voice controlled Tetris game based on Microsoft Speech recognition is described in detail.  Finally the system is tested and analyzed, it has  some practical value for its high recognition rate and its robustness. | A series of voice-related work is completed by COM  component and the speech recognition is managed by Recognition Engine. As a result, programmers only need to  concentrate on their own application development, call the  associated Speech Application Program Interface (SAPI) to  implement voice function. SAPI provides a high level interface  between applications and speech engine. Controlling and  management of various speech engines need real-time operation  technology. However, ASPI realizes and hides the underlying  technical detail. | Experiments show that, the voice controlled system lacks flexibility. |
| Towards Open Intent Discovery for Conversational Text | Existing research for intent discovery model it as a classification task with a predefined set of known categories. To generalize beyond these pre-existing classes, They have define a new task of open intent discovery. They investigate how intent can be generalized to those not seen during training. | They have proposed a two-stage approach, TOP-ID (Towards OPen Intent Discovery), aims to solve the problem of open intent discovery. In the first stage, their method employs a softmax classifier on top of a bidirectional LSTM to determine intent or not. If it does, the second stage of TOP-ID is applied to  identify and extract all possible intents in a consistent and generalizable problem.  They model this as a sequence tagging problem and solve it by developing a neural network model consisting of a Conditional Random Field (CRF) on top of a bidirectional LSTM, accompanied by a multi-head self-attention mechanism.  TOP-ID, can discover both previously seen as well as unseen (during training) user intents in diverse real-world scenarios.  It can identify multiple user intents per utterance.  We curate and present a large, intent-annotated dataset of 25K text instances from real-world task domains, without any restriction on the number or types of intents possible. | Doesn’t extract implicit intents. |
| Bot Colony – a Video Game Featuring Intelligent Language-Based Interaction with the Characters | This paper describes the dialog functionality available in the Beta version of Bot Colony, and discusses the applications of intelligent interactive conversation in  video games. Some advances in automated knowledge acquisition, entailment of dictionary definitions, paraphrase,  co-reference resolution to 3D scenes, spatial relations, and using natural language to program 3D animation –  enabling user-generated content - are presented. | The major components of our dialog pipeline are dialog management, parsing, disambiguation, co-  reference resolution, Question Answering (QA) and reasoning and generation.  The player can ask factual questions about location, time, objects, people, events, how to do something or why his interlocutor does something (R1 – R9). Answering questions such as ‘How do you know X?’ (R10, R15) and being environment-aware (see Reasoning) lends credibility to the  virtual characters.  The player can also ask robots to manipulate objects in various ways in order to advance in the game (R16, R21). A key game mechanic in Bot Colony is extracting information from a robot in exchange for helping it learn about humanity (the robot is motivated by the ‘do as people do’ law).  Player STATEMENTS are remembered by the robot. | Players statement are not checked for consistency with previous statements.  Players have to use well-formed English only.  Answers are computed dynamically, voice actors cannot be used with the technology described here. This introduces an artistic limitation. |
| Natural Language Processing and Game-based Practice in iSTART | Intelligent Tutoring Systems (ITSs) are situated in a potential struggle between effective pedagogy and system enjoyment and engagement. iSTART, a reading strategy tutoring system in which students practice generating self-explanations and using reading strategies, employs two devices to engage the user. The first is natural language processing (NLP). Incorporating NLP within iSTART allows students to use their own thoughts and ideas to communicate with the system, and serves as the core *intelligence* of the system that is used to drive the feedback and the adaptive interactions during practice. | Once the students are in the practice module, an animated character (Merlin) provides feedback on students’ explanations, prompting them to generate new explanations using their newly acquired  repertoire of strategies. The main focus of the practice module is to provide students with an opportunity to apply the reading strategies to new texts and to integrate their knowledge from different sources in order to understand a challenging text. Their  explanation may include world and domain knowledge or it may stem from prior sentences in the  text. Merlin provides feedback for each explanation generated by the student. For example, he may prompt them to expand the explanation, ask the students to incorporate more information, or  suggest that they make a connection back to other parts of the text. The iSTART algorithm is designed to assess the quality of the student’s response such that it can drive Merlin’s feedback  to the student in pedagogically effective ways.  The iSTART assessment algorithm evaluates each student self-explanation as a 0, 1, 2, or 3. An assessment of “0” relates to explanations that are either too short or contain mostly irrelevant information. An iSTART score of “1” is associated with an explanation that primarily relates only to the target sentence itself (sentence-based). A “2” means that the student’s explanation incorporated some aspect of the text beyond the target sentence (text-based). If an explanation earns a “3” from the iSTART evaluation, then the explanation incorporates information at a global level, and may include outside information or refer to an overall theme across the whole text (i.e., global-based information).  Their algorithm utilizes a combination of both word-based approaches and latent semantic analysis. Word-based approaches provide a lower level explanations (ones that are irrelevant, or simply repeat the target sentence), In contrast, LSA provides a higher level and more  complex explanations.  Students with initially low performance improved such that they were indistinguishable from the initially high performing students. Students who performed poorly on the self-explanation pretest, compared to students who performed well on the self-explanation pretest, produced significantly lower quality self-explanations on the first 10 texts, but not after having received sufficient training.  They developed an improvised version iSTART – ME. Main  goal of the iSTART-ME project was to integrate several game-based principles and features that are expected to support effective learning, increase motivation, and sustain engagement throughout a long-term interaction with an established ITS.  They used Mini-games model strategy and aimed to improve: identification of strategies, generation of new self-explanations, meta-comprehension  awareness, and/or vocabulary. Each mini-game focuses on one or two of these areas of  improvement, and situates it within a game-based environment. | They have not assessed their metrics against texts from other domains (other than science).  Studies further indicate that the text-based explanations (score of 2) are more difficult to distinguish by the iSTART algorithms.  Map Conquest (one of the mini game) was rated as significantly more frustrating than  the other generation games because it was confusing to the players.  iSTART – ME is primitive in terms of aesthetics, particularly in comparison to today’s gaming standards. |

**3. Conceptual System Design**

**3.1 Conceptual System Diagram:**

****

**3.2 Flowchart:**

**4. Implementation**

**4.1 CODE:**

import tkinter as tk

import random

import itertools

import requests

import sys

def get\_rand\_offset():

offset = [100,-100]

random.shuffle(offset)

return offset[0]

control\_lines = dict()

w = None

h = None

line\_lst = list()

def create\_grid\_and\_control\_lines(event=None):

global control\_lines, w, h, line\_lst

w = c.winfo\_width() # Get current width of canvas

h = c.winfo\_height() # Get current height of canvas

#c.delete('grid\_line') # Will only remove the grid\_line

# Creates all vertical lines at intevals of 100

for i in range(0, w, 100):

c.create\_line([(i, 0), (i, h)], tag='grid\_line')

# Creates all horizontal lines at intevals of 100

for i in range(0, h, 100):

c.create\_line([(0, i), (w, i)], tag='grid\_line')

choices = list(itertools.product(range(0, w - 100, 100), range(0, h - 100, 100)))

random.shuffle(choices)

j = 0

count = 0

colour = ['red','green','blue','gold']

random.shuffle(colour)

while j < 4:

origin\_x, origin\_y = choices[0]

end\_x, end\_y = -1, -1

random.shuffle(choices)

while(end\_x < 0 or end\_y < 0):

end\_x, end\_y = origin\_x + get\_rand\_offset(), origin\_y + get\_rand\_offset()

if({(origin\_x, origin\_y), (end\_x, end\_y)} in line\_lst):

print("fail")

continue

ID = c.create\_line([(origin\_x, origin\_y), (end\_x, end\_y)], tag = colour[j] + '\_line', fill = colour[j], width = 10)

line\_lst.append({(origin\_x, origin\_y), (end\_x, end\_y)})

control\_lines[colour[j] + '\_line'] = ID

j += 1

##print(c.find\_withtag("control\_line\_1"))

def rotate(ele):

print("hi")

ID = control\_lines[ele]

x1,y1,x2,y2 = c.coords(ID)

old\_coords = {(x1,y1),(x2,y2)}

if(y2 > y1):

y2 -= 100

y1 += 100

else:

y1 -= 100

y2 += 100

new\_coords = {(x1,y1),(x2,y2)}

if(new\_coords in line\_lst):

return

c.coords(control\_lines[ele], x1, y1, x2, y2)

line\_lst.remove(old\_coords)

line\_lst.append(new\_coords)

def translate(ele, direction, val):

ID = control\_lines[ele]

x1, y1, x2, y2 = c.coords(ID)

old\_coords = {(x1,y1),(x2,y2)}

disp = int(val) \* 100

if(direction == "up" and min(y1,y2) - disp >= 0 and {(x1, y1 - disp), (x2, y2 - disp)} not in line\_lst):

c.move(ID, 0, -disp)

elif(direction == "down" and max(y1,y2) + disp <= h and {(x1, y1 + disp), (x2, y2 + disp)} not in line\_lst):

c.move(ID, 0, disp)

elif(direction == "right" and max(x1,x2) + disp <= w and {(x1 + disp, y1), (x2 + disp, y2)} not in line\_lst):

c.move(ID, disp, 0)

elif(direction == "left" and min(x1,x2) - disp >= 0 and {(x1 - disp, y1), (x2 - disp, y2)} not in line\_lst):

c.move(ID, -disp, 0)

new\_coords = {(c.coords(ID)[0],c.coords(ID)[1]), (c.coords(ID)[2],c.coords(ID)[3])}

line\_lst.remove(old\_coords)

line\_lst.append(new\_coords)

def operate\_game\_by\_command(x):

try:

command = x

r = requests.get(f'https://centralindia.api.cognitive.microsoft.com/luis/v2.0/apps/67ec9d8b-fed1-4ded-a47d-c76f07d5ef93?staging=true&verbose=true&timezoneOffset=330&subscription-key=b95bd42248b548f5b94f34dad2bde47a&q=' + command)

result = r.json()

print(result)

except Exception as e:

print(f'{e}')

#result = {'query': 'move red line rightwards by 1 block and then rotate the blue line.', 'topScoringIntent': {'intent': 'move', 'score': 0.982636154}, 'intents': [{'intent': 'move', 'score': 0.982636154}, {'intent': 'rotate', 'score': 0.151299834}, {'intent': 'None', 'score': 0.00239448785}], 'entities': [{'entity': 'red', 'type': 'colour', 'startIndex': 5, 'endIndex': 7, 'score': 0.9873937}, {'entity': 'blue', 'type': 'colour', 'startIndex': 56, 'endIndex': 59, 'score': 0.998382449}, {'entity': 'rightwards', 'type': 'direction', 'startIndex': 14, 'endIndex': 23, 'score': 0.9956634, 'role': 'right'}, {'entity': '1', 'type': 'builtin.number', 'startIndex': 28, 'endIndex': 28, 'resolution': {'subtype': 'integer', 'value': '1'}}]}

intent = list()

temp = (requests.get(f'https://centralindia.api.cognitive.microsoft.com/luis/v2.0/apps/9add54d5-3a9a-4315-b153-daa4c53395ad?staging=true&verbose=true&timezoneOffset=330&subscription-key=b95bd42248b548f5b94f34dad2bde47a&q=' + command)).json()

flag = None

if(temp['topScoringIntent']['intent'] == 'MoveAndRotate'):

flag = True

for ele in result['intents']:

if(flag and (ele['intent'] == "move" or ele['intent'] == "rotate")):

intent.append(ele['intent'])

elif(ele['intent'] == "move" or ele['intent'] == "rotate"):

intent.append(ele['intent'])

break

else:

break

colour = list()

direction = list()

magnitude = list()

for ele in result['entities']:

if(ele['type'] == 'colour'):

colour.append(ele['role'])

elif(ele['type'] == 'direction'):

direction.append(ele['role'])

elif(ele['type'] == 'builtin.number'):

magnitude.append(ele['resolution']['value'])

print(intent,colour,direction,magnitude)

if(len(intent) == 0 or len(colour) == 0 or ('translate' in intent and (len(direction)== 0 or len(magnitude) == 0))):

num\_of\_ops = 0

else:

num\_of\_ops = max(len(intent), len(colour), len(direction), len(magnitude))

data = [intent, colour, direction, magnitude]

for i in range(num\_of\_ops):

if(data[0][0] == 'move'):

translate(data[1][0] + '\_line', data[2][0], data[3][0])

for spec in data:

if(len(spec) > 1):

del spec[0]

else:

rotate(data[1][0] + '\_line')

if(len(data[1]) > 1):

del data[1][0]

root = tk.Tk()

#win1 = tk.Toplevel(root)

c = tk.Canvas(root, bg='white')

root.wm\_attributes('-fullscreen','true')

c.pack(fill=tk.BOTH, expand=True)

win2 = tk.Toplevel(root)

win2.attributes("-topmost",True)

win2.geometry(str(root.winfo\_screenwidth()) + "x" + str(round(root.winfo\_screenheight()\*0.028)) + "+" + str(0) + "+" + str(round(root.winfo\_screenheight()\*0.938)))

e = tk.Entry(win2)

e.pack(fill=tk.BOTH, expand=True)

win\_img = tk.PhotoImage(file = 'win.ppm')

def callback(event):

global root,win\_img

if(e.get() != ''):

if any(wrd in (e.get().lower()).split(' ') for wrd in ['exit', 'quit', 'close']):

root.destroy()

sys.exit()

operate\_game\_by\_command(e.get())

# Check Win

x = set()

for ele in line\_lst:

x = x.union(ele)

if(len(x) == 4):

print("Won")

c.delete("all")

c.create\_image(round(root.winfo\_screenwidth()/2.3), round(root.winfo\_screenheight()/2.5), anchor = tk.NW, image = win\_img)

e.delete(0, 'end')

c.bind\_all('<Return>', callback)

c.bind('<Configure>', create\_grid\_and\_control\_lines)

##root.after(100,thread\_loop)

'''c.bind('<Button-1>', lambda event, ele = "red\_line":

rotate(ele))

c.bind\_all('<w>', lambda event, ele = "red\_line", direction = "up", val = 1:

translate(ele, direction, val))

c.bind\_all('<a>', lambda event, ele = "red\_line", direction = "left", val = 1:

translate(ele, direction, val))

c.bind\_all('<s>', lambda event, ele = "red\_line", direction = "down", val = 1:

translate(ele, direction, val))

c.bind\_all('<d>', lambda event, ele = "red\_line", direction = "right", val = 1:

translate(ele, direction, val))'''

e.focus\_set()

root.mainloop()

The code given above is a python code which shows how my game utilizes Tkinter to form its core GUI and LUIS’s intent and entity detection to get intents and parameters out of player’s natural language-based sentences. It also displays the operator function that converts player’s intents into a graphical form (Game GUI).

**4.2 OUTPUT:**

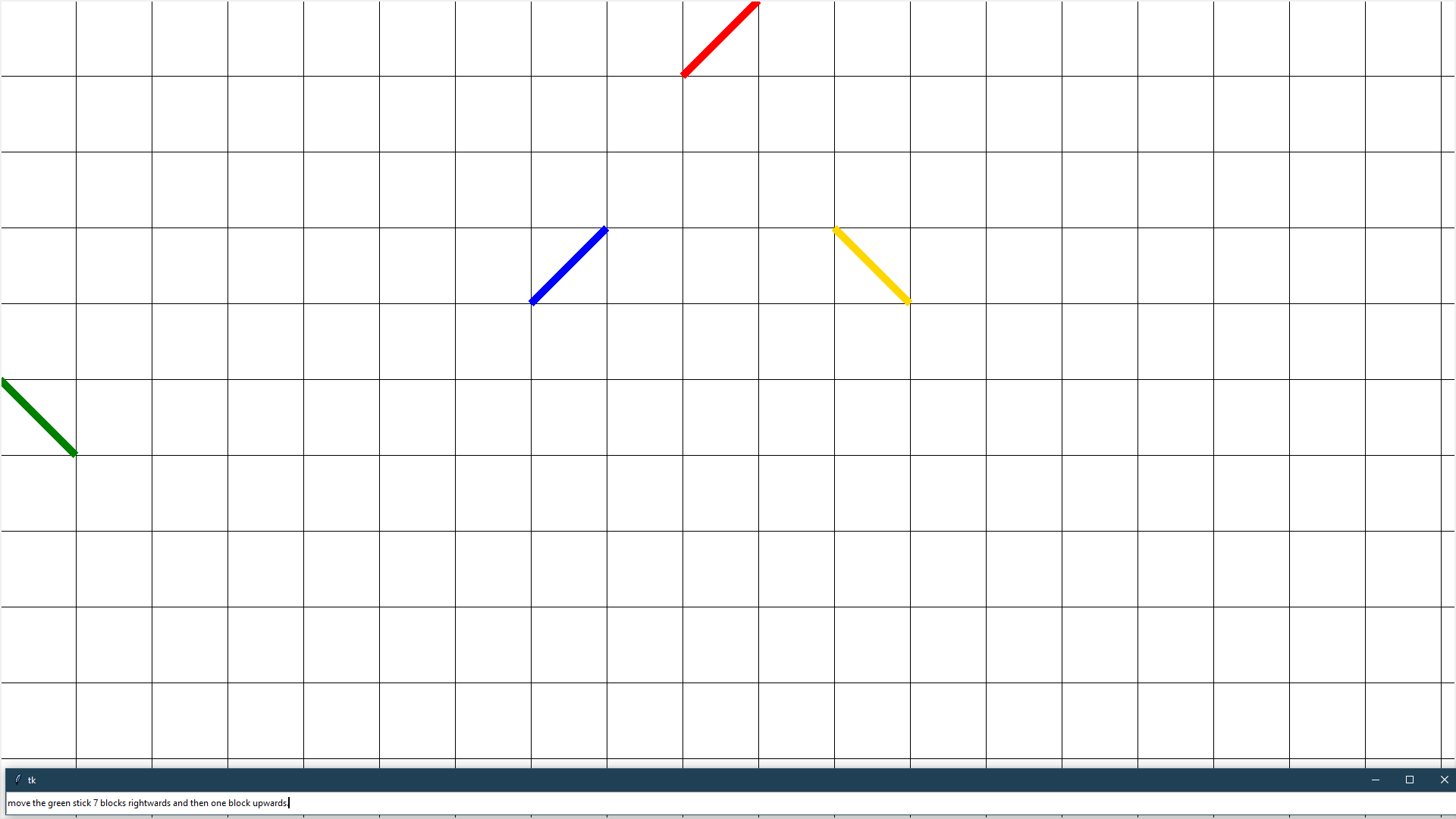
****

Image showing instruction to translate the green stick.

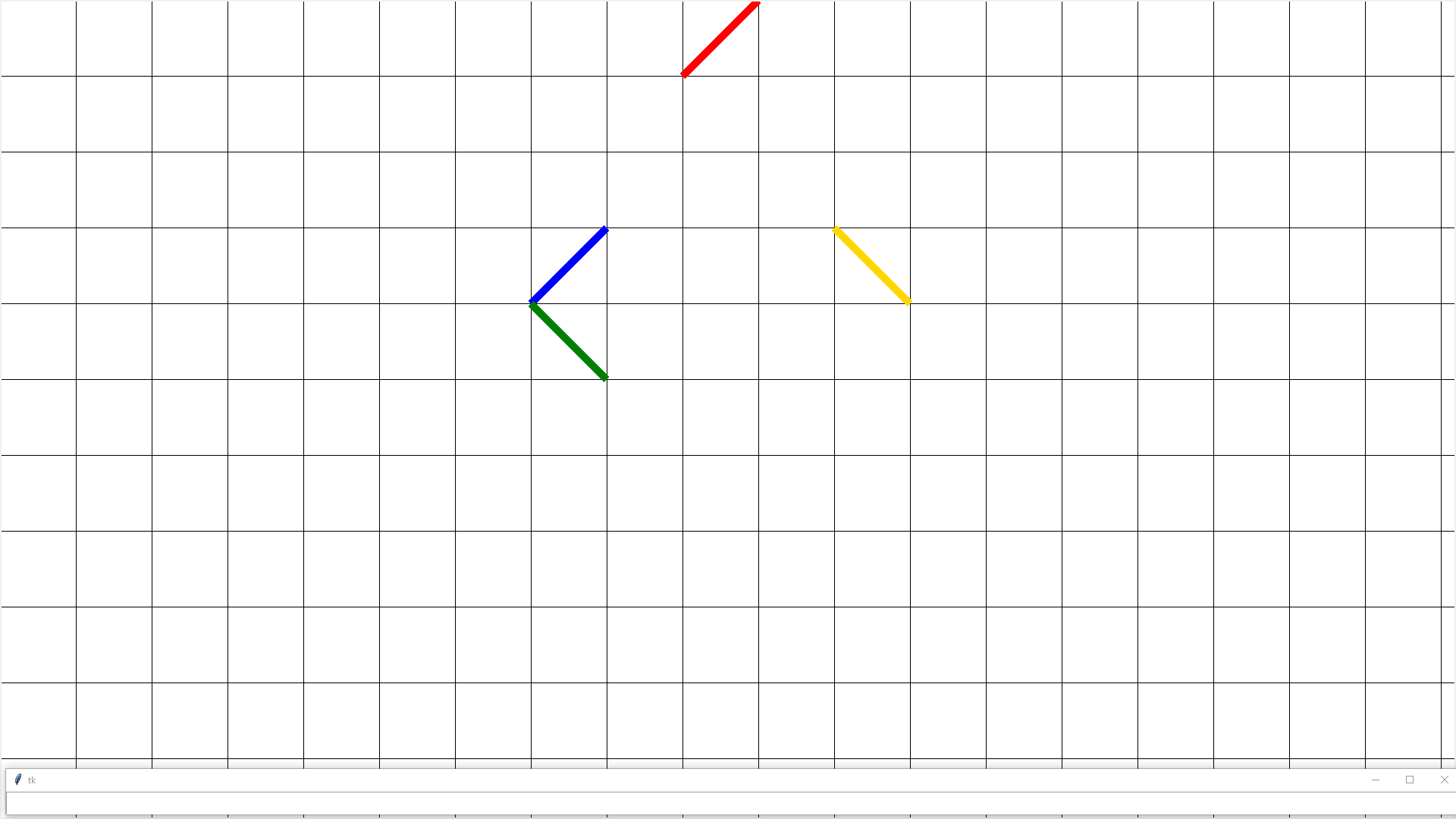
****

Image showing result of the above instruction.

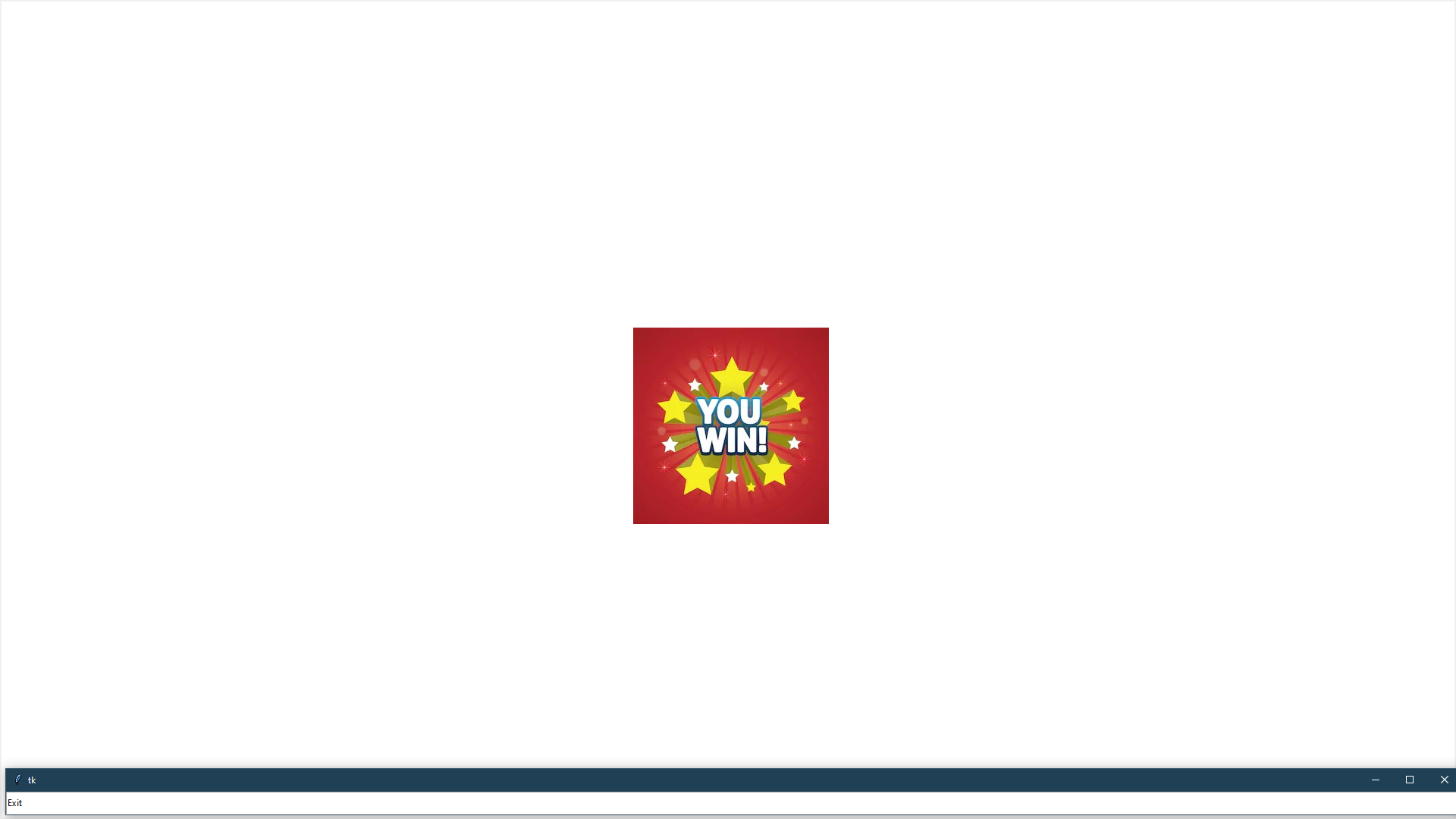
****

Image that is shown once the player wins the game by forming a diamond shape out of all the given sticks/lines.

**5. Conclusion and Future scope**

**5.1. Conclusion:**

Natura.IO attempts to increase the involvement of the player in the gameplay by controlling the game via natural language.

**5.2. Future scope:**

Voice controlled natural language game IO system, this will allow the player to shift his whole focus towards the game rather than on the input devices like keyboard, mouse, joystick, etc.

**6. References**

1. Nikam, Mohan, Pooja Dighe, Ruta Patel, Lalita Wagh and Varsha Yadav. “Voice Recognition Scheme for Coding Console Using Natural Language Processing-A Survey.” (2018).
2. Picca, Davide & Jaccard, Dominique & Eberlé, Gérald. (2015). Natural Language Processing in Serious Games: A state of the art.. International Journal of Serious Games. 2. 10.17083/ijsg.v2i3.87.
3. Zhu, Xiaodan & Penn, Gerald. (2006). Utterance-Level Extractive Summarization of Open-Domain Spontaneous Conversations with Rich Features. 793-796. 10.1109/ICME.2006.262600.
4. Jeswani, Hitesh. “A Study on Natural Language Processing for Human Computer Interaction.” (2017).
5. Joseph, E.: Bot Colony – a Video Game Featuring Intelligent Language-Based Interaction with the Characters (2014). https://www.botcolony.com/doc/BotColony\_paper.pdf
6. Jackson, T., Boonthum-Denecke, C., & McNamara, D. (2015). Natural language processing and game-based practice in iSTART. Journal of Interactive Learning Research, 26(2), 189-208.
7. M. Mehta and A. Corradini, "Evaluation of a domain independent approach to natural language processing for game-like user interfaces," 2009 IEEE Symposium on Computational Intelligence and Games, Milano, 2009, pp. 225-232.

doi: 10.1109/CIG.2009.5286470

1. Williams, Jason & Kamal, Eslam & Ashour, Mokhtar & Amr, Hani & Miller, Jessica & Zweig, Geoff. (2015). Fast and easy language understanding for dialog systems with Microsoft Language Understanding Intelligent Service (LUIS). 159-161. 10.18653/v1/W15-4622.
2. Yuan, XiaoJie & Fan, Jing. (2011). Design and implementation of voice controlled Tetris game based on Microsoft SDK. 10.1109/ICMT.2011.6001825.
3. Vedula, Nikhita & Lipka, Nedim & Maneriker, Pranav & Parthasarathy, Srinivasan. (2019). Towards Open Intent Discovery for Conversational Text.