Міністерство освіти і науки України
Національний технічний університет України «Київський політехнічний інститут імені Ігоря Сікорського»
Факультет інформатики та обчислювальної техніки

Кафедра інформатики та програмної інженерії

Звіт

з лабораторної роботи № 6 з дисципліни
«Проєктування алгоритмів»

„**Пошук в умовах протидії, ігри з повною інформацією, ігри з елементом випадковості, ігри з неповною інформацією**”

Виконав(ла) *ІП-35 Адаменко Арсен Богданович* _____
(шифр, прізвище, ім'я, по батькові)

Перевірив *Головченко М.М.* _____
(прізвище, ім'я, по батькові)

Київ 2024

ЗМІСТ

1 МЕТА ЛАБОРАТОРНОЇ РОБОТИ

Мета роботи - вивчити основні підходи до формалізації алгоритмів знаходження рішень задач в умовах протидії. Ознайомитися з підходами до програмування алгоритмів штучного інтелекту в іграх з повною інформацією, іграх з елементами випадковості та в іграх з неповною інформацією.

2 ЗАВДАННЯ

Для ігор з повної інформацією, згідно варіанту (таблиця 2.1) реалізувати візуальний ігровий додаток для гри користувача з комп'ютерним опонентом. Для реалізації стратегії гри комп'ютерного опонента використовувати алгоритм альфа-бета-відсікань. Реалізувати три рівні складності (легкий, середній, складний).

Для ігор з елементами випадковості, згідно з варіантом (таблиця 2.1) реалізувати візуальний ігровий додаток, з користувацьким інтерфейсом, не консольним, для гри користувача з комп'ютерним опонентом. Для реалізації стратегії гри комп'ютерного опонента використовувати алгоритм мінімакс або інший за потреби.

Для карткових ігор, згідно з варіантом (таблиця 2.1), реалізувати візуальний ігровий додаток, з користувацьким інтерфейсом, не консольним, для гри користувача з комп'ютерним опонентом. Потрібно реалізувати стратегію комп'ютерного опонента для гри з неповною інформацією.

Реалізувати анімацію процесу жеребкування (+1 бал) або реалізувати анімацію ігрових процесів (роздачі карт, анімацію ходів тощо) (+1 бал).

Реалізувати варто тільки одне з бонусних завдань.

Зробити узагальнений висновок лабораторної роботи.

Таблиця 2.1 – Варіанти

№	**Варіант**	**Тип гри**
1	Яцзи https://game-wiki.guru/published/igryi/yaczzyi.html	З елементами випадковості
2	Лудо http://www.iggamecenter.com/info/ru/ludo.html	З елементами випадковості
3	Генерал http://www.rules.net.ru/kost.php?id=7	З елементами випадковості
4	Нейтріко	З повною

	http://www.iggamecenter.com/info/ru/neutreeko.html	інформацією
5	Тринадцять http://www.rules.net.ru/kost.php?id=16	З елементами випадковості
6	Індійські кості http://www.rules.net.ru/kost.php?id=9	З елементами випадковості
7	Dots and Boxes https://ru.wikipedia.org/wiki/Палочки_(игра)	З повною інформацією
8	Двадцять одне http://gamerules.ru/igry-v-kosti-part8#dvadtsat-odno	З елементами випадковості
9	Тіко http://www.iggamecenter.com/info/ru/teeko.html	З повною інформацією
10	Клоббер http://www.iggamecenter.com/info/ru/clobber.html	З повною інформацією
11	101 https://www.durbetsel.ru/2_101.htm	Карткові ігри
12	Hackenbush http://www.papg.com/show?1TMP	З повною інформацією
13	Табу https://www.durbetsel.ru/2_taboo.htm	Карткові ігри
14	Заєць і Вовки (за Зайця) http://www.iggamecenter.com/info/ru/foxh.html	З повною інформацією
15	Свої козирі https://www.durbetsel.ru/2_svoi-koziri.htm	Карткові ігри
16	Війна з ботами https://www.durbetsel.ru/2_voina_s_botami.htm	Карткові ігри
17	Domineering 8x8 http://www.papg.com/show?1TX6	З повною інформацією
18	Останній гравець https://www.durbetsel.ru/2_posledny_igrok.htm	Карткові ігри
19	Заєць и Вовки (за Вовків) http://www.iggamecenter.com/info/ru/foxh.html	З повною інформацією
20	Богач https://www.durbetsel.ru/2_bogach.htm	Карткові ігри
21	Редуду https://www.durbetsel.ru/2_redudu.htm	Карткові ігри

22	Эльферн https://www.durbetsel.ru/2_elfern.htm	Карткові ігри
23	Ремінь https://www.durbetsel.ru/2_remen.htm	Карткові ігри
24	Реверсі https://ru.wikipedia.org/wiki/Реверси	З повною інформацією
25	Вари http://www.iggamecenter.com/info/ru/oware.html	З повною інформацією
26	Яцзи https://game-wiki.guru/published/igryi/yaczzyi.html	З елементами випадковості
27	Лудо http://www.iggamecenter.com/info/ru/ludo.html	З елементами випадковості
28	Генерал http://www.rules.net.ru/kost.php?id=7	З елементами випадковості
29	Сим https://ru.wikipedia.org/wiki/Сим_(игра)	З повною інформацією
30	Col http://www.papg.com/show?2XLY	З повною інформацією
31	Snort http://www.papg.com/show?2XM1	З повною інформацією
32	Chomp http://www.papg.com/show?3AEA	З повною інформацією
33	Gale http://www.papg.com/show?1TPI	З повною інформацією
34	3D Noughts and Crosses 4 x 4 x 4 http://www.papg.com/show?1TND	З повною інформацією
35	Snakes http://www.papg.com/show?3AE4	З повною інформацією

3 ВИКОНАННЯ

3.1 Програмна реалізація алгоритму

3.1.1 Вихідний код

```
from random import randint
from copy import deepcopy
import tkinter as tk

categories = [
    '1', '2', '3', '4', '5',
    # 'S3', 'S4',
    # 'S23',
    # 'L4', 'L5',
    # 'Y',
    # 'C'
]

def eval_score(dices, category):
    counts = {d: 0 for d in range(1, 5 + 1)}

    for d in dices.values():
        counts[d] += 1

    if category == '1':
        return counts[1] * 1

    elif category == '2':
        return counts[2] * 2

    elif category == '3':
        return counts[3] * 3

    elif category == '4':
        return counts[4] * 4

    elif category == '5':
        return counts[5] * 5

    elif category == 'S3':
        return sum(dices) if max(counts.values()) ⩾ 3 else 0

    elif category == 'S4':
        return sum(dices) if max(counts.values()) ⩾ 4 else 0

    elif category == 'S23':
        return 25 if sorted(counts.values(), reverse=True)[:2] == [3, 2] else
0

    elif category == 'L4':
        straights = [
            {1, 2, 3, 4},
            {2, 3, 4, 5},
            {3, 4, 5, 6}
        ]
```

```
        return 30 if any(s ⩽ set(dices.values()) for s in straights) else 0

    elif category == 'L5':
        straights = [
            {1, 2, 3, 4, 5},
            {2, 3, 4, 5, 6}
        ]
        return 40 if any(s ⩽ set(dices.values()) for s in straights) else 0

    elif category == 'Y':
        return 50 if max(counts.values()) == 5 else 0

    elif category == 'C':
        return sum(dices.values())

    return 0

cats_usage = {
    c: False for c in categories
}
dices = {
    1: 1,
    2: 2,
    3: 3,
    4: 3,
    5: 3
}

def gen_keeps():
    keeps = []

    for i in range(32):
        bins = list(map(int, f'{bin(i):0<7}'[2:][::-1]))

        keeps += [{i + 1: bins[i] for i in range(0, 5)}]

    return keeps

def gen_keep():
    return {i: bool(randint(0, 1)) for i in range(1, 5 + 1)}

def gen_rolled(dices, keep):
    new_dices = deepcopy(dices)

    for k in dices:
        if not keep[k]:
            new_dices[k] = randint(1, 5)

    return new_dices

def eval_randomly(dices, cats_usage, deepness, start_keep):
    cats_usage = deepcopy(cats_usage)

    for iter_ in range(deepness):
        if all(cats_usage.values()):
            if iter_ == 0:
                return {'score': 0, 'cat': 'C'}
```

```
            return max_score_info

        scores_info = [{'score': eval_score(dices, cat), 'cat': cat} for cat
in categories if not cats_usage[cat]]

        max_score_info = max(scores_info, key=lambda si: si['score'])

        cats_usage[max_score_info['cat']] = True

        if iter_ == 0:
            new_keep = start_keep
        else:
            new_keep = gen_keep()

        dices = gen_rolled(dices, new_keep)

    return max_score_info

def best_move(dices, cats_usage, rolls):
    reses = {}
    trials = 1000
    for keep in gen_keeps():
        key = ''.join([str(int(d)) for d in keep.values()])
        reses[key] = 0

        for _ in range(trials):
            reses[key] += eval_randomly(dices, cats_usage, rolls - 1, keep)
['score']

        reses[key] /= trials

    fmt_move = max(reses, key=lambda k: reses[k])

    return {i + 1: v for i, v in enumerate(map(int, fmt_move))}

no_keep = {i + 1: False for i in range(5)}
start_dices = {i + 1: i + 1 for i in range(5)}

def bot_make_turn():
    global bot_cats_label
    global bot_cats_scores
    global bot_cats_usage
    global no_keep
    global start_dices
    global bot_dices_label
    global is_game_end

    if is_game_end:
        return

    dices = gen_rolled(start_dices, no_keep)

    for i in range(3):
        if i == 2:
            break

        move = best_move(dices, bot_cats_usage, 3 - i)
```

```
        calced_keep = move

        dices = gen_rolled(dices, calced_keep)

        bot_dices_label.config(text = 'Bot Dices: ' + ' '.join(str(i) for i in
dices.values()))

    best_cat = None
    best_cat_score = -1
    for cat in categories:
        cur_score = eval_score(dices, cat)

        if cur_score > best_cat_score and not bot_cats_usage[cat]:
            best_cat_score = cur_score
            best_cat = cat

    bot_cats_usage[best_cat] = True
    bot_cats_scores[best_cat] = best_cat_score
    bot_cats_label.config(text = 'Bot: ' + ' '.join(str(i) for i in
bot_cats_scores.values()))
    bot_dices_label.config(text = 'Bot Dices: ' + ' '.join(str(i) for i in
dices.values()))

    print(bot_cats_scores)
    print(dices)

    end_game()

def roll_event():
    global you_roll_n
    global you_dices
    global input_keep_entry
    global dices_label
    global is_game_end

    if is_game_end:
        return

    if you_roll_n ⩾ 3:
        return

    keep_fmt = input_keep_entry.get()

    keep = {i + 1: False for i in range(5)}

    for v in map(int, keep_fmt.split()):
        keep[v] = True

    you_dices = gen_rolled(you_dices, keep)

    dices_label.config(text = 'Dices: ' + ' '.join(str(i) for i in
you_dices.values()))

    you_roll_n += 1

def use_cat_event():
    global you_roll_n
    global you_dices
```

```
    global you_cats_usage
    global you_cats_label
    global you_cats_scores
    global input_cat_entry
    global is_game_end

    if is_game_end:
        return

    end_game()

    if you_roll_n == 0:
        return

    cat = input_cat_entry.get()

    if you_cats_usage[cat]:
        return

    you_cats_scores[cat] = eval_score(you_dices, cat)
    you_cats_label.config(text = 'You: ' + ' '.join(str(i) for i in
you_cats_scores.values()))
    you_cats_usage[cat] = True

    you_roll_n = 0

    bot_make_turn()

    end_game()

def end_game():
    global win_label
    global you_cats_usage
    global bot_cats_usage
    global you_cats_scores
    global bot_cats_scores
    global is_game_end

    if not (all(you_cats_usage.values()) and all(bot_cats_usage.values())):
        return

    you_scores = sum((0 if not isinstance(you_cats_scores[k], int) else
you_cats_scores[k]) for k in you_cats_scores)
    bot_scores = sum((0 if not isinstance(bot_cats_scores[k], int) else
bot_cats_scores[k]) for k in bot_cats_scores)

    print(you_scores)
    print(bot_scores)

    if you_scores > bot_scores:
        text = 'You wins!'
    elif bot_scores > you_scores:
        text = 'Bot wins!'
    else:
        text = 'Tie!'

    win_label.config(text = text)
```

```
    is_game_end = True

you_roll_n = 0
you_dices = deepcopy(dices)

is_game_end = False

you_cats_usage = {
    c: False for c in categories
}
you_cats_scores = {
    c: 'x' for c in categories
}
bot_cats_usage = {
    c: False for c in categories
}
bot_cats_scores = {
    c: 'x' for c in categories
}

win = tk.Tk()

title_label = tk.Label(
    win,
    text = 'Simulator of melalchoholic Yathzee high-definition very realistic
simulator'
)

cats_label = tk.Label(
    win,
    text = 'Categories: 1 2 3 4 5 S3 S4 S23 L4 L5 Y C'
)

you_cats_label = tk.Label(
    win,
    text = 'You: x x x x x x x x x x x'
)

bot_cats_label = tk.Label(
    win,
    text = 'Bot: x x x x x x x x x x x'
)

bot_dices_label = tk.Label(
    win,
    text = 'Bot Dices: x x x x x'
)

dices_label = tk.Label(
    win,
    text = 'Dices: x x x x x'
)

input_cat_entry = tk.Entry(
    win,
    bg = 'lightgreen'
)
```

```
input_keep_entry = tk.Entry(
    win,
    bg = 'yellow'
)

roll_button = tk.Button(
    win,
    text = 'Roll',
    command = roll_event
)

use_cat_button = tk.Button(
    win,
    text = 'Use',
    command = use_cat_event
)

win_label = tk.Label(
    win,
    text = 'Game continues ... '
)

title_label     .grid(row=0, column=0, rowspan=1, columnspan=2)
cats_label      .grid(row=1, column=0, rowspan=1, columnspan=2)
you_cats_label  .grid(row=2, column=0, rowspan=1, columnspan=2)
bot_cats_label  .grid(row=3, column=0, rowspan=1, columnspan=2)
bot_dices_label .grid(row=4, column=0, rowspan=1, columnspan=2)
dices_label     .grid(row=5, column=0, rowspan=1, columnspan=2)
input_cat_entry .grid(row=6, column=0, rowspan=1, columnspan=2)
input_keep_entry.grid(row=7, column=0, rowspan=1, columnspan=2)

roll_button     .grid(row=8, column=0, rowspan=1, columnspan=1)
use_cat_button  .grid(row=8, column=1, rowspan=1, columnspan=1)

win_label       .grid(row=9, column=0, rowspan=1, columnspan=2)

win.mainloop()
```

3.1.2 Приклади роботи

На рисунках 3.1 і 3.2 показані приклади роботи програми.

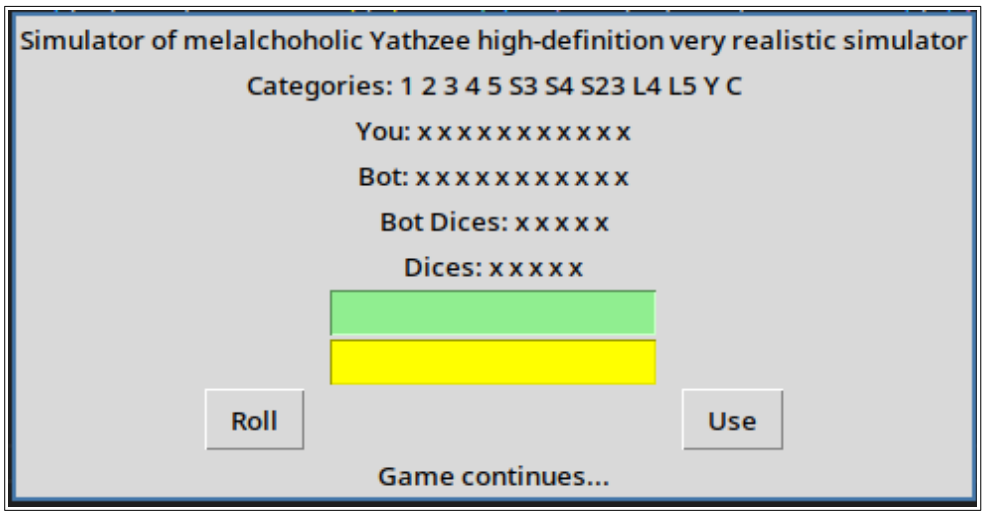

Рисунок 3.1 – вікно гри

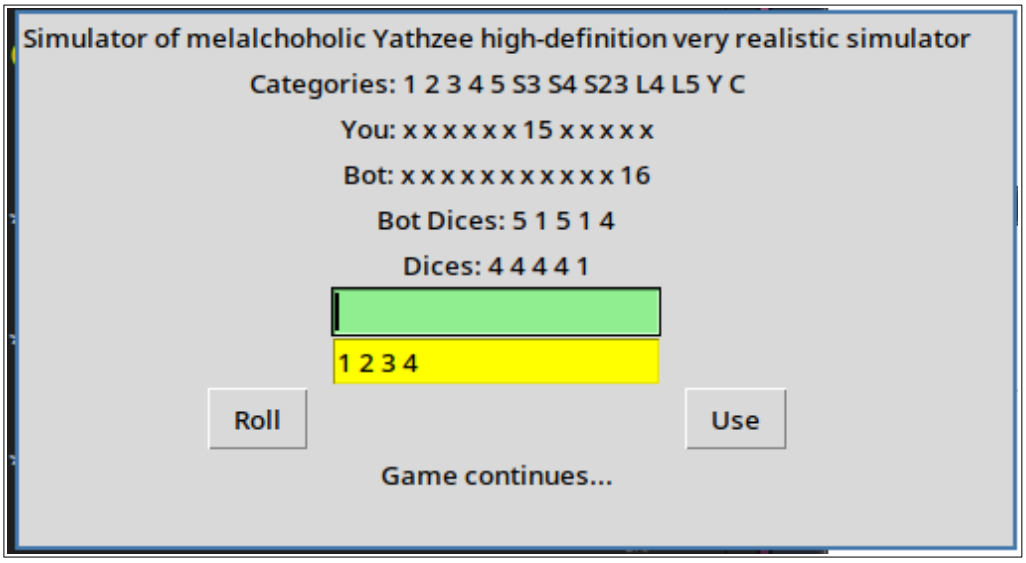

Рисунок 3.2 – типова ситуація в Яцзи №1

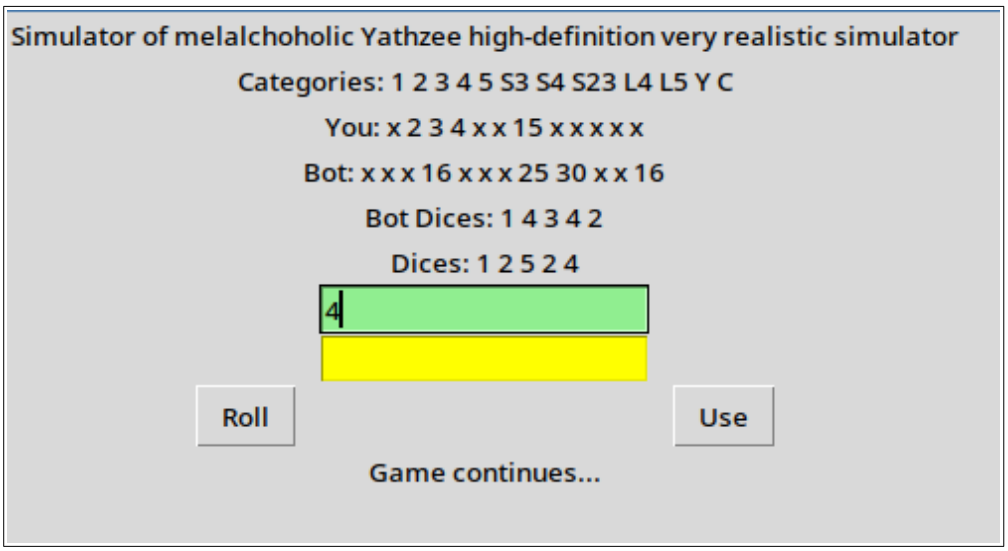

Рисунок 3.3 – типова ситуація в Яцзи №2

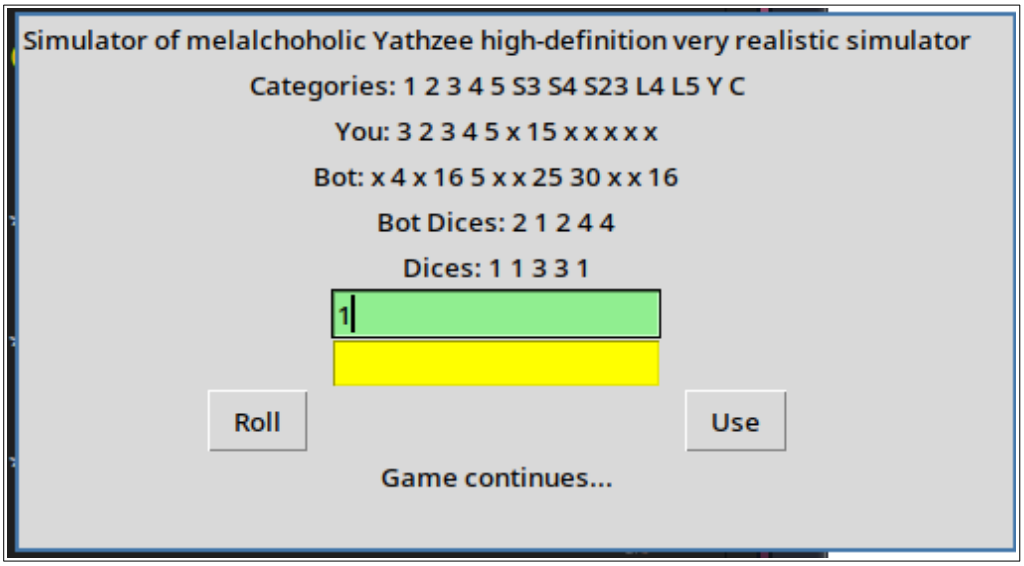

Рисунок 3.4 – типова ситуація в Яцзи №3

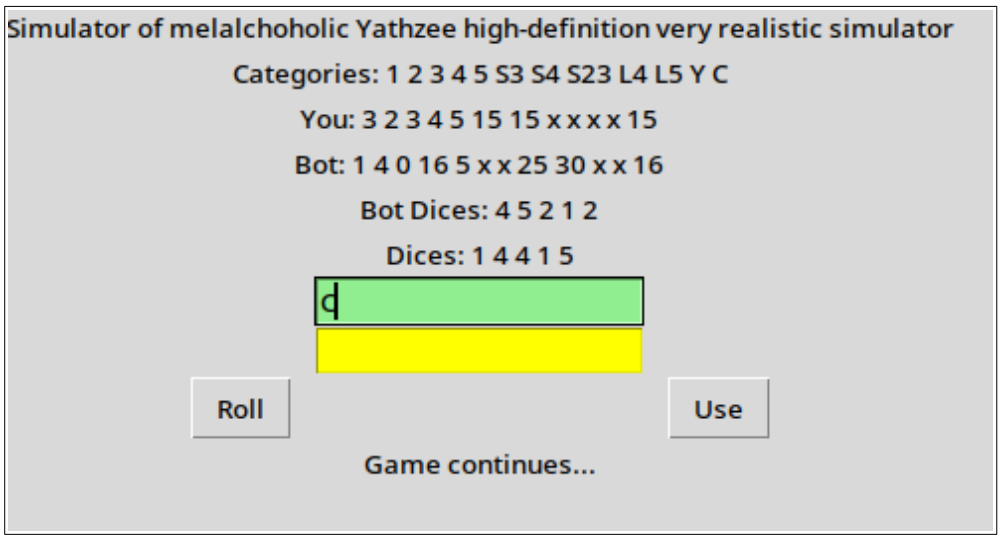

Рисунок 3.4 – типова ситуація в Яцзи №4

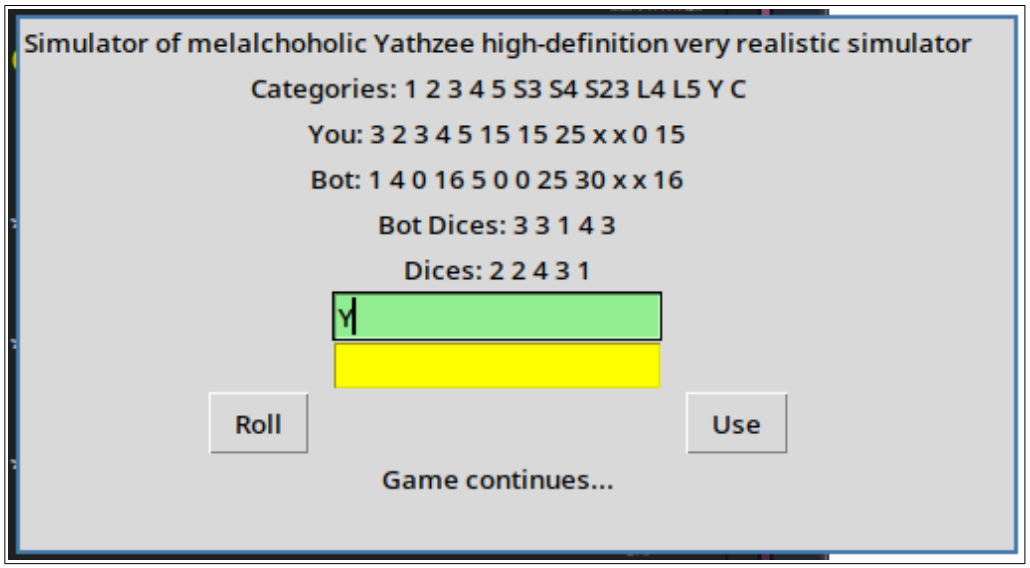

Рисунок 3.5 – типова ситуація в Яцзи №5

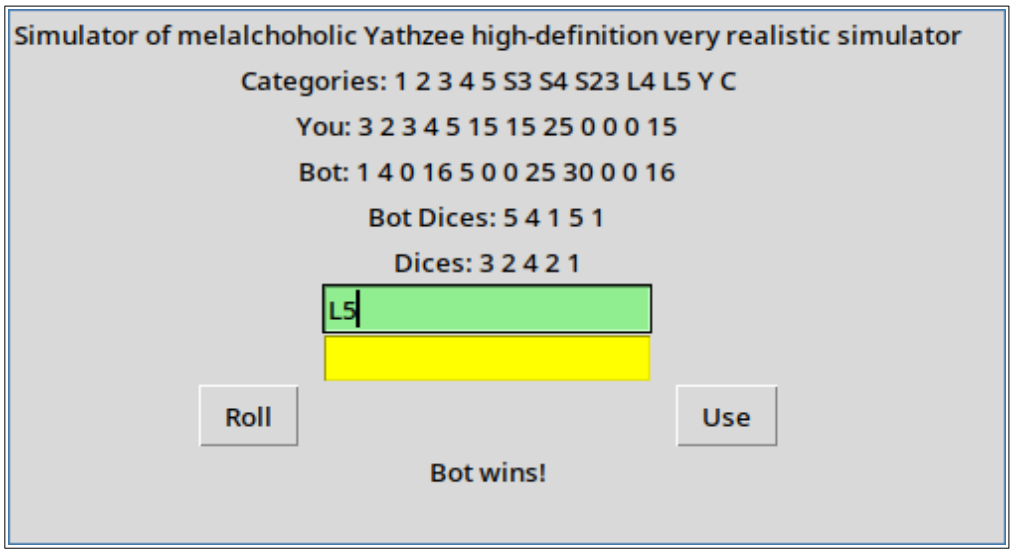

Рисунок 3.6 – перемога одного з гравців

ВИСНОВОК

В рамках даної лабораторної роботи я нарешті зміг реалізувати гру Яцзи з елементиами випадковсті використовуючи алгоритми знаходження найкращого рішення або цінності серед усіх можливих ходів в поточний момент часу.

КРИТЕРІЇ ОЦІНЮВАННЯ

Критерії оцінювання у відсотках від максимального балу:

- програмна реалізація – 95%;
- висновок – 5%.

+1 додатковий бал можна отримати за реалізацію анімації ігрових процесів (жеребкування, роздачі карт, анімацію ходів тощо).