Politechnika Wrocławska

Wydział Elektroniki, Fotoniki i Mikrosystemów

Projektowanie Algorytmów i Metody Sztucznej Inteligencji

Kółko i krzyżyk

Projekt 3 - zadanie na ocenę bdb

Prowadzący:
Dr inż. Łukasz Jeleń

Wykonała:
Zuzanna Mejer, 259382

Wrocław, 8 czerwca 2022r.

Spis treści

1 Wprowadzenie **2**

2 Opis tworzonej gry **2**

3 Graficzny interfejs użytkownika **2**
3.1 Kółko i Krzyżyk 2
3.2 Rysowanie planszy 3
3.3 Obsługa „wydarzeń” 3
3.4 Oznaczenie wygranej 4

4 Techniki AI **4**

5 Podsumowanie i wnioski **4**

6 Literatura **5**

1. Wprowadzenie

Zadanie polegało na zaimplementowaniu gry Kółko i Krzyżyk z wykorzystaniem technik sztucznej inteligencji - algorytmu Minimax z alfa-beta cięciami. Użytkownik miał mieć możliwości definiowania rozmiaru planszy wraz z ilością znaków w rzędzie.

2. Opis tworzonej gry

Kółko i Krzyżyk to gra o sumie zerowej, w której bierze udział dwóch użytkowników. W tym przypadku jest to gracz i komputer. Obydwoje gracze dążą do ustawienia swoich znaków w rzędzie, kolumnie lub po przekątnej tak, aby zachować ciągłość swoich znaków (żeby znak przeciwnika nie przeszkodził w tworzeniu linii). Jednocześnie, gracze dążą do tego, żeby przerwać ciągłość znaków przeciwnika w linii. W celu stworzenia gry w Kółko i Krzyżyk przyjęto parę założeń:

- Grę zawsze rozpoczyna Kółko.
- Użytkownik nie może wybrać czy jest Kółkiem, czy Krzyżykiem.Użytkownik zawsze jest Kółkiem i zatem zawsze rozpoczyna grę.
- Przed rozpoczęciem gry użytkownik musi zdefiniować rozmiar kwadratowej planszy, to znaczy ile chce mieć kratek w rzędzie.
- W zależności od wybranego rozmiaru planszy, do wygranej należy ułożyć w rzędzie, kolumnie lub po przekątnej tyle znaków, ile wynosi liczba kratek w jednej linii na planszy. To znaczy, że trzeba ze znaków ułożyć ciągłą linię od jednego końca planszy, do drugiego.

3. Graficzny interfejs użytkownika

W celu ułatwienia użytkownikowi obsługi gry, stworzono graficzny interfejs. Wykorzystano w tym celu bibliotekę programistyczną *SFML* (Simple and Fast Multimedia Library).

3.1. Kółko i Krzyżyk

Kółko i Krzyżyk zostały zaimplementowane i rysowane jako litery „X” i „O” czcionką Arial o odpowiednio dobranym rozmiarze tak, żeby wpasowywały się w środek kratki, niezależnie od liczby kratek na planszy. Przedstawia to poniższy kod:

```
1     sf::Font font;
2     if (!font.loadFromFile("arial.ttf"))
3     {
4         std::cout << "Nie mamy takiej czcionki\n";
5     }
6
7     sf::Text circle;
8     circle.setFont(font);
9     circle.setString("O");
10    circle.setCharacterSize(one_cell_size);
11    circle.setFillColor(sf::Color::Black);
```

```
12      circle.setLineSpacing(0);
13      circle.setLetterSpacing(0);
14
15
16      sf::Text cross;
17      cross.setFont(font);
18      cross.setString("X");
19      cross.setCharacterSize(one_cell_size);
20      cross.setFillColor(sf::Color::Black);
21      cross.setLineSpacing(0);
22      cross.setLetterSpacing(0);
```

3.2. Rysowanie planszy

Rysowanie planszy odbywa się tylko raz na początku gry. Po podaniu przez użytkownika rozmiaru planszy, program dostosowuje rozmiar jednej kratki do wielkości okna, zdefiniowanej jako zmienna globalna: *constexpr int one_side_number_of_cells = 800;*. Następnie rysuje kratki za pomocą prostokątów o odpowiednim rozmiarze i odpowiedniej rotacji. Jest to przedstawione poniżej:

```
1  void draw_board(sf::RenderWindow &window, int cells)
2  {
3      int one_cell_size = window.getSize().x / cells;
4      int tmp = one_cell_size;
5      sf::RectangleShape rectangle(sf::Vector2f(5, one_side_number_of_cells));
6      rectangle.setFillColor(sf::Color::Black);
7
8      for (int i = 1; i <= cells - 1; i++)
9      {
10         rectangle.setPosition(one_cell_size, 0);
11         window.draw(rectangle);
12         one_cell_size = one_cell_size + tmp;
13     }
14
15     one_cell_size = window.getSize().x / cells;
16     rectangle.rotate(-90);
17     for (int i = 1; i <= cells - 1; i++)
18     {
19         rectangle.setPosition(0, one_cell_size);
20         window.draw(rectangle);
21         one_cell_size = one_cell_size + tmp;
22     }
23 }
```

3.3. Obsługa „wydarzeń”

Każdorazowe kliknięcie myszką na planszę jest obsługiwane przez *sf::Event*. Po wykryciu kliknięcia, obliczana jest pozycja myszki i rysowany jest znak kółka w odpowiednim miejscu. Niestety program

nie jest odporny na błędy w postaci kilkukrotnego klikania myszką w jedno miejsce. W takim przypadku na planszy pojawi się więcej krzyżyków tak, jakby gra toczyła się dalej. Kod przedstawiony poniżej opisuje także obsługę zamknięcia okna z grą, co jest jednoznaczne z zakończeniem gry.

```
while (board.pollEvent(event))
{
    if (event.type == sf::Event::Closed)
        board.close();

    if (event.type == sf::Event::MouseButtonPressed)
    {
        if (event.mouseButton.button == sf::Mouse::Left)
        {
            sf::Vector2i position = sf::Mouse::getPosition(board);
            int column = position.x / one_cell_size;
            int row = position.y / one_cell_size;
            tab[row][column] = 'O';
            circle.setPosition(column * one_cell_size +
            (circle.getCharacterSize() * 0.1),
            row * one_cell_size + (circle.getCharacterSize() * -0.13));
            board.draw(circle);
            board.display();
        }
    }
}
```

3.4. Oznaczenie wygranej

ohoho

4. Techniki AI

5. Podsumowanie i wnioski

1. o tych kliknieciach ze nie ok kilkukrotne
2. no czy dziala To
3. i dla jakich glebokosci
4. ze sfml jest super, ale szkoda ze nie ma buttons
5. gdybym robila dopracowanie to jeszcze bym dodala mozliwosci 2: 1. wybranie przez uzytkownika kolko czy krzyzyk i 2. moze ile tych znakow w rzedzie do wygranej itd.

6. Literatura

- https://www.sfml-dev.org/ (dostęp: 8.06.2022)
- https://en.wikipedia.org/wiki/Minimax (dostęp: 8.06.2022)
- Notatki z wykładów