Projekt Systemu Informatycznego  
  
  
  
  
  
  
„Wizualizacja Sieci Neuronowej Na Podstawie Gry 2D”

Autor: Mikołaj Jasiński 5IGRS

Prowadzący: mgr Jacek Żywczok

Wyższa Szkoła Technologii Informatycznych Katowice 2022

18.01.2023

# Spis Treści

[Wstęp 3](#_Toc124916048)

Cel projektu

Motyw

Wstępna ilustracja

[Założenia projektowe 4](#_Toc124916049)

Gra

Symulacja

Wizualizacja

[Ograniczenia projektowe 5](#_Toc124916050)

[Chronologiczny plan pracy 6](#_Toc124916051)

[Narzędzia i technologie 7](#_Toc124916052)

[Wymagania niefunkcjonalne 7](#_Toc124916053)

Sprzętowe

Systemowe

[Opis najważniejszych komponentów 8](#_Toc124916054)

Sieć neuronów

Player

FishPlayer

Przeszkody

Sensory

Wizualizacja

[Podręcznik użytkownika 11](#_Toc124916055)

Przeznaczenie programu

Menu

Tryb solo

Symulacja

[Przebieg uruchomiania programu 15](#_Toc124916056)

[Testowanie projektu 16](#_Toc124916058)

Środowiska testowe

Dane wejściowe użytkownika

Znane błędy

[Podsumowanie 17](#_Toc124916059)

[Spis ilustracji 18](#_Toc124916060)

[Bibliografia 19](#_Toc124916061)

# Wstęp

Cel projektu

Celem projektu jest zwizualizowanie użytkownikowi jednokierunkowej sieci neuronowej na podstawie akcji pełnionych przez wirtualnego agenta grającego w grę 2D.

Aplikacja pozwala użytkownikowi zapoznać się prosty sposób jak działają sieci neuronowe na prostym przykładzie zastosowania ich grze 2D.

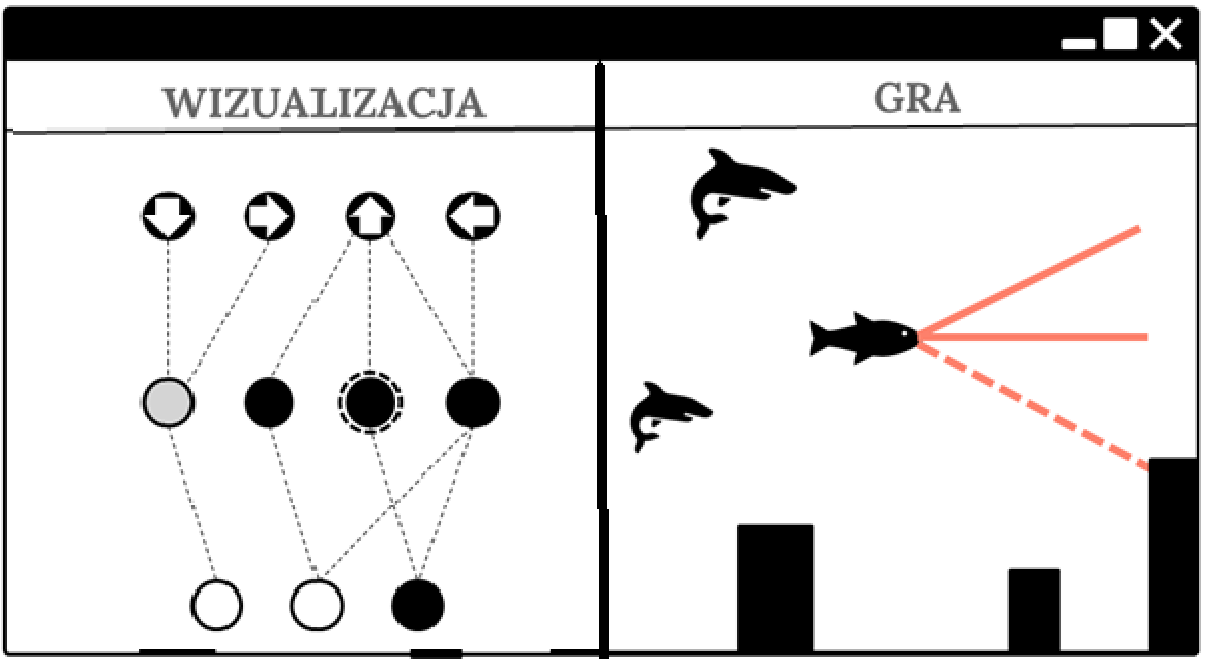
Motyw

W informatyce siecią neuronową określa się oprogramowanie, zbliżone w działaniu do neuronów w ludzkim mózgu.

Wyróżniającą cechą sieci neuronowej jako narzędzia informatycznego jest możliwość komputerowego rozwiązywania przy jej pomocy praktycznych problemów bez ich uprzedniej matematycznej formalizacji.

Dla większości osób trudno wyobrazić sobie jak działa sieć neuronów, stąd motywacja aby stworzyć projekt. Jako dodatkowe motywacje to zgłębienie osobistej wiedzy na temat sieci neuronowych oraz języku programowania C#

Wstępna ilustracja



Rysunek 1 - Wstępna ilustracja

# Założenia projektowe

Gra

Gra będzie opierała się na postaci poruszającej się z lewej do prawej strony ekranu z nieskończonym przewijaniem.

Na drodze będą pojawiały się przeszkody, których należy unikać, aby dalej poruszać się do przodu w grze, tym samym zwiększając wynik.

Im dłużej postać będzie omijała przeszkody, tym wyższy będzie oceniany wynik.

Gracz będzie kontrolował postać mechanizmem kontrolowania składającym się z czterech klawiszy (W, S, D i A).

Użytkownik będzie miał opcje grania w grę samemu – w celu zapoznania się z grą.

Symulacja

Wirtualny agent grający w grę będzie posiadał określoną przez użytkownika liczbę sensorów wychodzących z postaci które służyć będą jako wejścia do sieci neuronowej.

Kiedy linia sensora wejdzie w kontakt z przeszkodą jego wartość wejściowa ulegnie zmianie, co będzie zasygnalizowane zmianą koloru sensora.

Wartości wejściowe z sensorów będą służyć jako wejścia do sieci neuronowej (wartością wejść będzie odległość przeszkody od sensora), wyjściami będzie mechanizm kontrolowania postaci.

Wizualizacja

Wizualizacja będzie opierała się na okręgach odpowiadającym wejściami i wyjściami, oraz liniami odpowiadającymi połączeniami neuronów.

Okręgi będą miały różne kolory na podstawie stronniczości (wartości powyżej której neuron jest aktywny).

Linie będą mniej lub bardziej widoczne na podstawie ich wagi (jak duży wpływ dane wejście ma na wyjście).

Okręgi będą podkreślane kiedy odpowiadające im wyjścia i wejścia są aktywne.

Sieci neuronowe będą automatycznie generowane losowo ze startem symulacji.

W celu zwiększenia interaktywności symulacji użytkownik aplikacji będzie mógł zmieniać parametry symulacji, takie jak liczba postaci naraz grających w grę (wizualizacja będzie pokazywała sieć neuronową wybranej postaci) oraz parametry sieci neuronowej takie jak liczba „ukrytych” neuronów oraz liczba sensorów na postaci.

# Ograniczenia projektowe

* Sieć neuronowa na której będzie opierała się wizualizacja będzie prostą siecią jednokierunkową z małą liczbę wejść, wyjść i warstw.
* Grafika będzie wielce uproszczona.
* Rozdzielczość okna gry będzie stałą wartością.
* Użytkownik będzie limitowany co do wpływu na parametry z jakimi startuję symulację.
* Sieci neuronowe będą generowane kompletnie losowo.
* Z racji użycia Windows Forms, gra nie będzie cross-platformowa

# Chronologiczny plan pracy

1. Stworzenie okna pozwalającego użytkownikowi na nawigację po aplikacji oraz standardowe guziki kontroli okna (minimalizacja, zamknięcie aplikacji).
2. Stworzenie mechanizmu który będzie rysował i wyświetlał świat 2D w sposób pozwalający na nieskończone przewijanie z lewej do prawej strony na podstawie framework’u MonoGame.
3. Stworzenie prostej postaci oraz dodanie mechanizmu tworzenia przeszkód.
4. Implementacja mechanizmu kontroli nad postacią (przyśpieszanie, zwalnianie, poruszanie w górę i w dół), w sposób który pozwala na przekonywujący wygląd poruszania się postaci po świecie gry.
5. Implementacja mechanizmu zliczającego wynik na podstawie odległości przebytej przez postać.
6. Dodanie mechanizmu kończonego grę i pokazującego wynik gdy postać uderzy w przeszkodę.
7. Dodanie sensorów w formie linii i implementacja zmiany ich wyglądu i wartości gdy linia spotka się z przeszkodą.
8. Zaprogramowanie mechanizmu generującego prostą sieć neuronową i procesu pozwalającego na przekazywanie wartości z sensorów jako wejść do sieci.
9. Dodanie przekazywania kontroli do sieci przez podłączenie wyjść do mechanizmu kontrolowania postaci.
10. Dodanie procesu który będzie pozwalał na jednoczesne granie wielu postaci kontrolowanych przez wirtualnych agentów w tym samym czasie
11. Stworzenie wizualizacji wygenerowanej sieci tak aby znajdowała się ona po lewej stronie okna.
12. Stylizacja oraz dodanie animacji dla wizualizacji.
13. Dodanie mechanizmu który służy do pokazywania wizualizacji dla postaci wybranej przez użytkownika
14. Implementacja dodatkowych funkcji pozwalających użytkownikowi na zmiany parametrów startowych symulacji w menu.

# Narzędzia i technologie

* Visual Studio 2022 IDE
* Framework .NET Core
* Język programowania C#
* Framework Monogame
* Framework Windows Forms

# Wymagania niefunkcjonalne

Sprzętowe

* Komputer obsługujący system operacyjny Windows
* Monitor o rozdzielczości przynajmniej 1324x768 pikseli.
* Przynajmniej 20 MB wolnej pamięci na dysku.
* Przynajmniej 100 MB wolnej pamięci RAM.
* Klawiatura i myszka.

Systemowe

* Przynajmniej Windows 7
* Zainstalowane oprogramowanie .NET 6.0

# 

# Opis najważniejszych komponentów

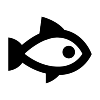
Sieć neuronów

Liczba wejść odpowiada liczbie sensorów, liczba wyjść odpowiada czterema możliwościami kontroli postaci (do góry, w doł, w lewo, w prawo). Natomiast liczba neuronów w warstwie ukrytej jest parametrem kontrolowanym przez użytkownika.

Wartości sieci neuronów (stronniczość, wagi połączeń) są generowane przypadkowo.

Decyzja o tym czy dany neuron jest aktywny decyduje algorytm FeedForward.

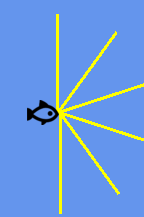
Player



Rysunek 2 - Tekstura Player

Player to postać kontrolowana przez gracza. Może poruszać się w prawo, w lewo, do góry lub w dół w zależności od przytrzymanego klawisza.

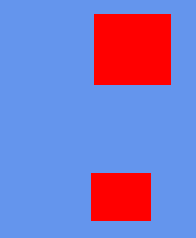
FishPlayer



Rysunek 3 – FishPlayer

FishPlayer rozszerza klasę Player. Jest to postać w symulacji kontrolowana przez sieć neuronów (każdy FishPlayer ma inną sieć neuronów)

Przeszkody

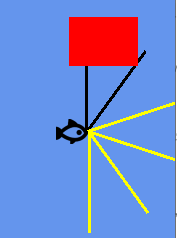


Rysunek 4 – przeszkody

Podczas gry w trybie solo lub symulacji generowane są przeszkody z przypadkowymi parametrami oraz pozycją które poruszają się z daną prędkością z prawej do lewej strony.

Gdy ryba (FishPlayer, Player) spotka się z przeszkodą to przegrywa.

Sensory



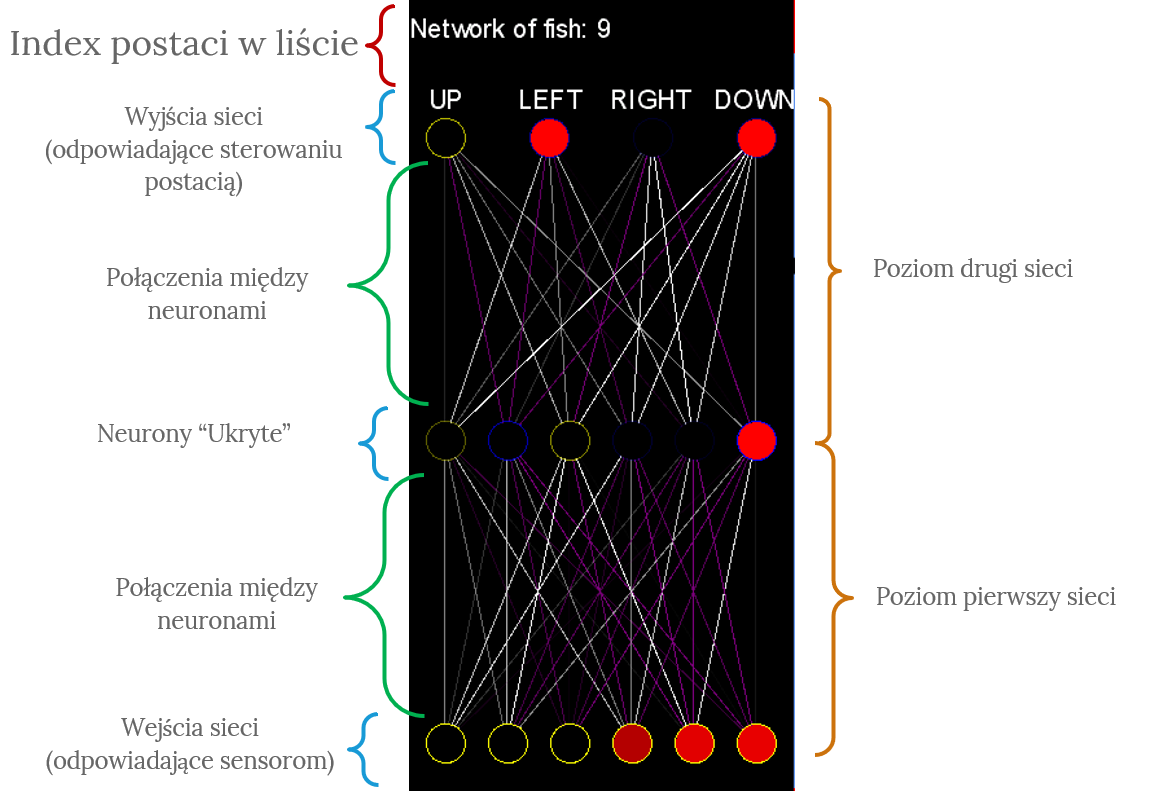
Rysunek 5 - Sensory

Każda ryba (SmartPlayer) podczas symulacji ma przynajmniej 2 sensory, kiedy linia sensora przecina się z przeszkodą zmienia ona swój kolor z żółtego na czarny i odległość przecięcia od początku linii wysyła jako wejście do sieci neuronowej od ryby do której należy.

To pod jakim kątem generowane są poszczególne sensory decyduje algorytm interpolacji liniowej.

Wizualizacja

Wizualizacja to element rysowany z lewej strony ekranu podczas symulacji.



Rysunek 6 - Wizualizacja

Kolor czerwony w neuronach oznacza że są one aktywne – czym mniejsza transparentność koloru tym większa wartość na neuronie. Kolor obramowania neuronów odpowiada czy stronniczość danego neuronu jest negatywna czy pozytywna (niebieski – negatywna, zółty pozytywna) natomiast transparentność obramowania odpowiada wartości bezwzględnej stronniczości (czym większa tym mniej transparentna).

Połączenia między neuronami kolorowane są na niebiesko jeżeli ich wagi są negatywne lub na biało jeżeli są pozytywne. Transparentność linii decyduje wartość bezwzględna wagi połączenia (czym większa tym mniej transparentna).

To gdzie na czarnym tle rysowane są poszczególne sensory decyduje algorytm interpolacji liniowej.

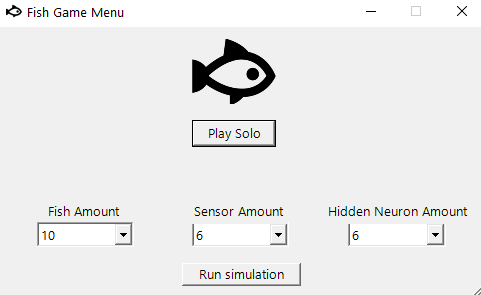
# Podręcznik użytkownika

Przeznaczenie Programu

Program jest przeznaczony na urządzenia z systemem Windows. Pozwala na zwizualizowanie sieci neuronowej generowanej przypadkowo w kontekście gry 2D.

Menu

Po starcie programu ukazuje się menu „Fish Game Menu”

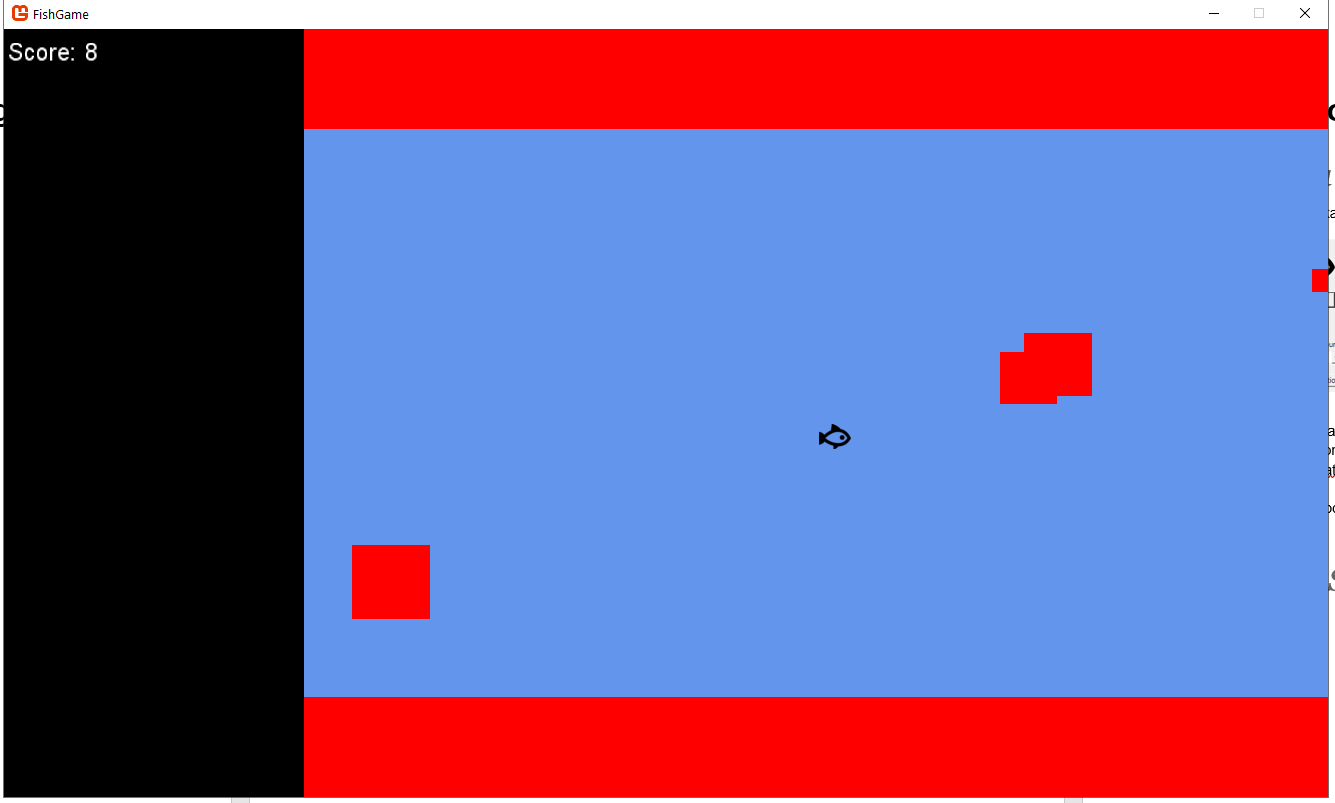


Rysunek 7 - Menu

Użytkownik może wybrać z 3 list dostępne parametry kontrolujące liczbę startujących ryb, liczbę sensorów i liczbę neuronów na ukrytym poziomie sieci neuronowej każdej z ryb przed kliknięciem „Run simulation” – guzik ten rozpoczyna symulację z parametrami wybranymi przez użytkownika.

Guzik „Play Solo” rozpoczyna grę w trybie solo gdzie użytkownik może zapoznać się z sterowaniem i grą.

Tryb Solo



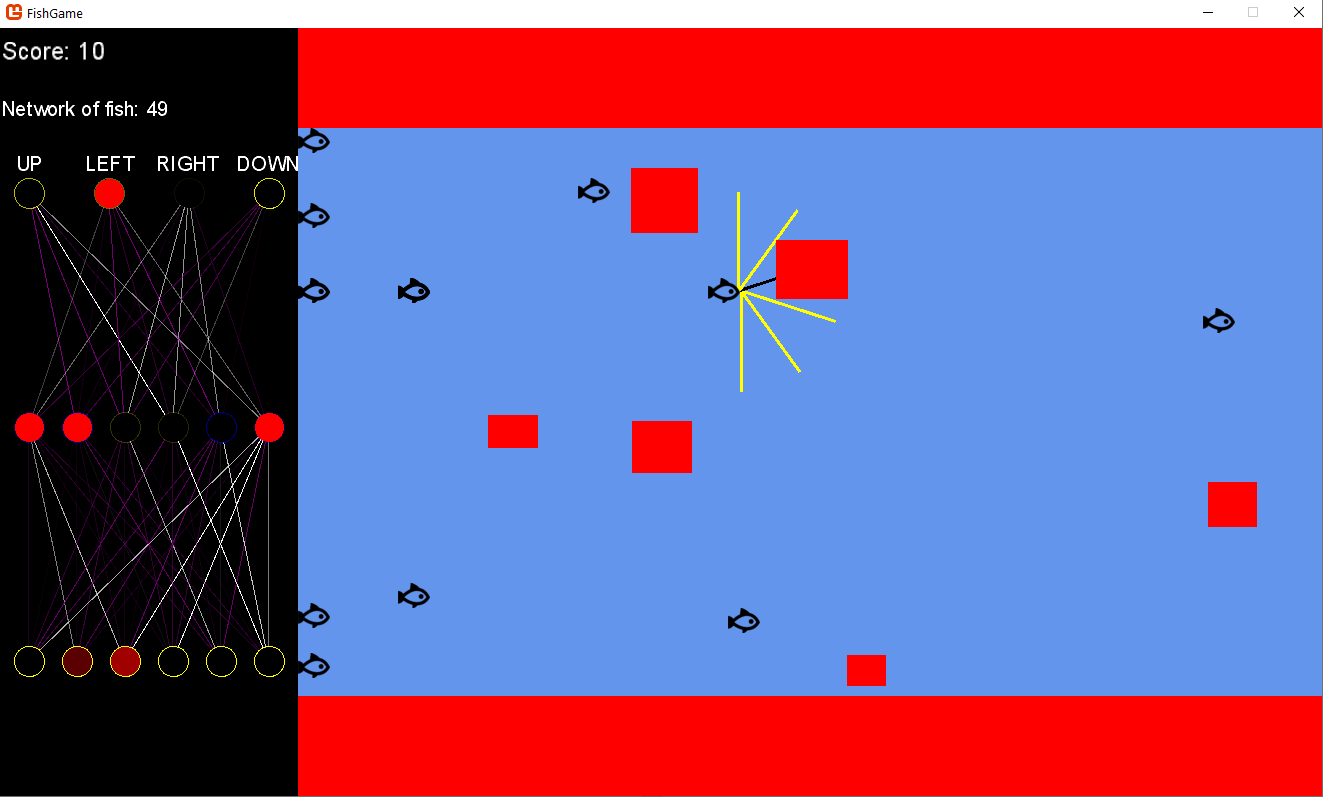
Rysunek 8 – Tryb Solo

W trybie solo użytkownik kontroluje poruszanie się ryby klawiszami W – w górę, S – w dół, D – w prawo, A – w lewo.

W lewym górnym rogu naliczają się punkty wynikające z czasu przetrwania bez dotykania czerwonych prostokątów.

Gra dobiega końca gdy użytkownik najedzie rybą na przeszkodę – wtedy zostaje wyświetlany w nowym oknie wynik, po kliknięciu OK w nowym oknie użytkownik wraca do menu.

Symulacja



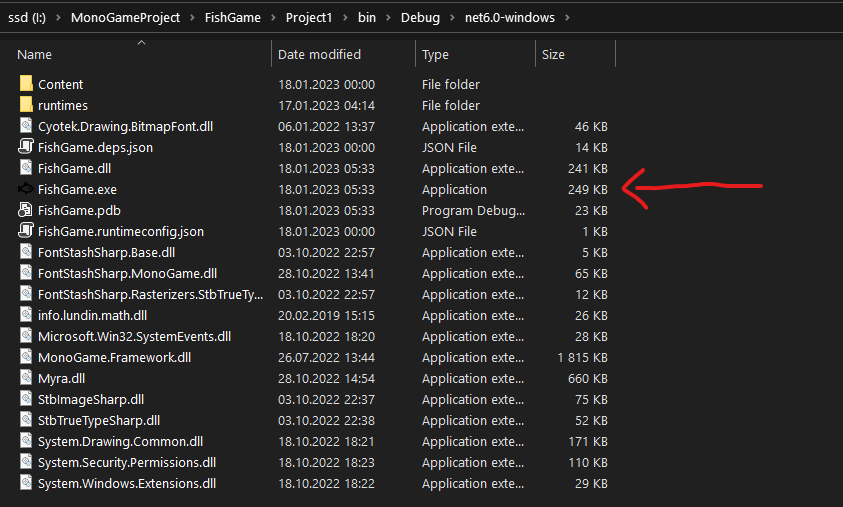
Rysunek 9 - Symulacja

W trybie symulacji gra generuje podaną wcześniej liczbę ryb – każda z własnymi sensorami oraz siecią neuronową. Po lewej stronie użytkownikowi ukazuje się wizualizacja sieci neuronowej danej ryby, użytkownik może przewijać między rybami używając klawiszy strzałka w lewo oraz strzałka w prawo. Gdy wszystkie ryby przegrają symulacja kończy się wyświetlając okno z wynikiem, po kliknięciu OK program wraca do menu.

# 

# Przebieg uruchomiania programu

Aby uruchomić program należy kliknąć dwukrotnie na FishGame.exe



Rysunek 10 - Uruchamianie programu

Lub jeżeli chcemy uruchomić w Visual Studio należy kliknąć „Start without debugging” po otwarciu pliku FishGame.snl

# 

Rysunek 11 - Uruchamianie programu w VS

# Testowanie projektu

Środowiska testowe

* Windows 10 – program zachowuje się poprawnie.
* Windows 11 – program zachowuje się poprawnie.

Dane wejściowe użytkownika

Program zachowuje się poprawnie z wszystkimi wartościami jakie użytkownik może wybrać w menu.

Znane błędy

* Przeszkody znikają za czarną granicą w trybie gry solo
* Z racji niezmiennej rozdzielczości okna gry na dużych ekranach okno jest bardzo małe
* Przesuwanie okna gry z ekranu o większej rozdzielczości na ekran o mniejszej rozdzielczości wprowadza artefakty graficzne w oknie gry.
* Na niektórych systemach okazjonalnie po zamknięciu okna gry wyskakuje okno błędu.
* Przy dużej liczbie sensorów oraz ryb na wolniejszych systemach gra może okazjonalnie się zacinać

# Podsumowanie

# Spis ilustracji

[Rysunek 1 - Wstępna ilustracja 3](#_Toc124916534)

[Rysunek 2 - Tekstura Player 8](#_Toc124916535)

[Rysunek 3 – FishPlayer 8](#_Toc124916536)

[Rysunek 4 – przeszkody 9](#_Toc124916537)

[Rysunek 5 - Sensory 9](#_Toc124916538)

[Rysunek 6 - Wizualizacja 10](#_Toc124916539)

[Rysunek 7 - Menu 11](#_Toc124916540)

[Rysunek 8 – Tryb Solo 12](#_Toc124916541)

[Rysunek 9 - Symulacja 13](#_Toc124916542)

[Rysunek 10 - Uruchamianie programu 14](#_Toc124916543)

[Rysunek 11 - Uruchamianie programu w VS 14](#_Toc124916544)

# Bibliografia

Dokumentacja Monogame Framework:

<https://docs.monogame.net/>

<https://community.monogame.net/t/how-to-draw-a-filled-circle-in-monogame/18062>

Tutorial Monogame:

<https://github.com/Oyyou/MonoGame_Tutorials>

<https://github.com/MonoGame/MonoGame.Samples>

Video sieci neuronowe:

<https://www.youtube.com/watch?v=aircAruvnKk>

<https://www.youtube.com/watch?v=aeWmdojEJf0>

Artykuły o implementacji sieci neuronowej w C#:

<https://towardsdatascience.com/building-a-neural-network-framework-in-c-16ef56ce1fef>

<https://rubikscode.net/2022/07/04/implementing-simple-neural-network-in-c/>

C# Interpolacja liniowa:

<https://stackoverflow.com/questions/12838007/c-sharp-linear-interpolation>

C# Geometria:

<https://stackoverflow.com/questions/5514366/how-to-know-if-a-line-intersects-a-rectangle>

<https://math.stackexchange.com/questions/404407/new-x-coordinate-of-a-rotated-line>

<https://stackoverflow.com/questions/2083942/draw-the-two-lines-with-intersect-each-other-and-need-to-find-the-intersect-poin>

Windows Forms:

<https://learn.microsoft.com/en-us/dotnet/desktop/winforms/?view=netdesktop-6.0>

<https://stackoverflow.com/questions/9648381/how-to-prevent-manual-input-into-a-combobox-in-c-sharp>

Projekty inspiracyjne:

<https://github.com/ssusnic/Machine-Learning-Flappy-Bird>

<https://www.youtube.com/watch?v=qv6UVOQ0F44>

<https://www.youtube.com/watch?v=zIkBYwdkuTk>

<https://github.com/greerviau/SnakeAI>

<https://github.com/gniziemazity/Self-driving-car>

Stack Overflow:

<https://stackoverflow.com/questions/56851555/monogame-level-depth-in-various-draw-functions>

<https://stackoverflow.com/questions/166802/windows-form-designer-could-not-load-file-or-assembly>

<https://stackoverflow.com/questions/32084424/c-sharp-monogame-passing-arguments-on-startup>

[https://stackoverflow.com/questions/12764544/xna-with-windows-forms-opening-new-window](https://stackoverflow.com/questions/12764544/xna-with-windows-forms-opening-new-window%09)

Ikona ryby:

<https://www.klipartz.com/es/sticker-png-cndzu>