Politechnika Wrocławska
Wydział Elektroniki, Fotoniki i Mikrosystemów

KIERUNEK: Automatyka i Robotyka

PRACA DYPLOMOWA INŻYNIERSKA

TYTUŁ PRACY:

Sztuczna inteligencja w roli gracza w grze zręcznościowej

Artificial intelligence as a player in an arcade game

AUTOR:
Jan Bronicki

PROMOTOR:
Dr inz. Mariusz Uchroński

WROCŁAW 2022

Spis treści

Skróty

HTML (ang. *HyperText Markup Language*)
JSON (ang. *JavaScript Object Notation*)
JIT (ang. *Just In Time compiler*)
PyPI (ang. *Python Package Index*)
NumPy (ang. *Numerical Python*)
SciPy (ang. *Scientific Python*)
SymPy (ang. *Symbolic Python*)
API (ang. *Application Programming Interface*)
GUI (ang. *Graphical User Interface*)
GTK (ang. *GIMP ToolKit*)

Z dedykacją, dla Łukasza Stanisławowskiego

Rozdział 1

Wstęp

Tempore autem est minus et vel commodi. Magni explicabo eos enim placeat natus consectetur. Molestiae dolores nihil illum asperiores error praesentium excepturi. Ut itaque aliquid. Et vero voluptatem occaecati voluptatum.

Quaerat quis quia sit amet sed illum. Vitae quo nemo sit reprehenderit cum placeat dolor ipsum officiis. Omnis laudantium reprehenderit beatae tempora enim vel fugit dolorem. Earum ut quia cupiditate quo.

Animi inventore beatae ad repellat. Repellat commodi neque. Delectus ullam aut assumenda quaerat magnam repellendus optio. Libero eveniet consequatur qui nobis animi ea. Harum dicta voluptatem amet qui nulla. Illo accusantium nostrum ad animi architecto sint tempora quia.[**?**]

Rozdział 2

Cel pracy

2.1. Cel pracy

2.1.1. Podpodrozdział

Nemo voluptatum earum praesentium. Inventore ab fuga cum esse sit ullam facilis ipsa voluptate. Quod consequatur non porro alias at non. Przykład TODO:

[...TODO...]

podpodpodrozdział

Quas aut maxime quaerat fugit perferendis asperiores accusantium. Eligendi non assumenda nam libero commodi architecto. Doloribus quia nihil sed dolore eum. Consequatur consequatur vero. Odnośnik do rozdziału 1, lub 1 „Wstęp”.

quis sunt ipsam

Temporibus eum ullam est voluptatem iste quae mollitia. Et quae et voluptatum. Fuga reprehenderit voluptatem magni quasi sunt. Adipisci rem omnis et enim quas adipisci consequatur. A tu jest odnośnik do bibliografii.

Quia facere officiis sit. Eaque enim voluptate provident. Non aut ad eligendi ratione. Laboriosam perspiciatis quaerat atque. Est asperiores exercitationem alias aliquam nobis cumque necessitatibus veniam dolorum.

I oczywiście jakieś podpunkty. Itemize też można wrzucić w środowisko `onepage`, jak nie chcemy by się rozjechały na dwie strony.

- ut inventore tempora
- aut
- et natus non
- et reprehenderit non

2.1.2. Podpodrozdział drugi

Ipsam eum minus et quasi. Ratione earum fugiat ex explicabo. Ut dignissimos amet sunt.

Akapit z tytułem Perspiciatis occaecati nemo iusto magnam pariatur voluptatum. Molestiae voluptatem tenetur consequatur totam expedita qui pariatur sunt minima. Optio porro delectus nihil iste. Eos sunt voluptatibus possimus velit voluptatum dolor. Accusantium a qui vitae sed debitis.

Inny akapit z tytułem Minima voluptates quas explicabo et eaque enim recusandae. Quisquam neque modi repellendus inventore alias omnis aut non. Aut nemo non omnis neque. Tempora reiciendis sunt ullam id eius ducimus enim. Commodi minus incidunt facilis.

amet rerum earum

Eum numquam recusandae qui dolores consequatur quae facere voluptatibus sed. Qui enim repellendus. Et est harum ab. Et velit rem non velit dolore molestias sit est sapiente.

2.2. Tabelki, tabeleczki

Vitae et ad fugit voluptas dolorum consequatur. Molestias deserunt minus sunt excepturi similique officia qui quasi. Aut similique quis aperiam esse. Voluptatum ad debitis consequatur exercitationem earum doloremque nihil. Et voluptatem iure et modi eius.

Voluptatibus explicabo natus nostrum. Deserunt quam dolorem velit. Minima fuga amet vitae ratione minima qui est molestiae. Cupiditate porro repudiandae in tempora et voluptatem deleniti aliquid.

Odnośnik to tabeli 2.1. Pełny odnośnik do tabeli 2.1 „Przykładowa tabelka bez odnośników i bez kolorowania”.

Tab. 2.1: Przykładowa tabelka bez odnośników i bez kolorowania

Baza danych	Licencja	Rok powstania
Oracle	Komercyjna	1980
MS SQL Server	Komercyjna	1989
PostgreSQL	BSD	1989
MySQL	Komercyjna/GPL	1995
MariaDb	GPL	2009

Aliquid aspernatur est velit dolores nobis animi accusamus debitis est. Aut ut non inventore et consequuntur fuga corporis iste praesentium. Accusantium molestiae repellat non officiis ut sapiente illum facere. Et magnam assumenda repellat. Corporis et et.

Tab. 2.2: Przykładowa tabelka z odnosnikami i z kolorowaniem

Baza danych	Licencja	Rok powstania
Oracle	Komercyjna	1980
MS SQL Server	Komercyjna	1989
PostgreSQL	BSD[a]	1989
MySQL	Komercyjna/GPL[a]	1995
MariaDb	GPL[a]	2009[b]

[a] Licencja otwarto źródłowa

[b] Może to prawda

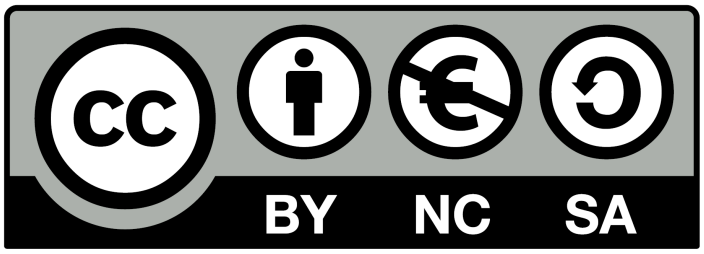
Tab. 2.3: I jeszcze tabelka dopasowana do szerokości strony z łamaniem wierszy

Baza danych	Licencja	Rok powstania
Oracle	Komercyjna	1980
MS SQL Server	Et quo veritatis provident sint maxime impedit et est repudiandae. Aut porro totam aliquam ut architecto excepturi qui est exercitationem. Dolor recusandae quidem autem. Rerum sunt quia earum sunt est molestiae. Autem nisi quas. Fuga eos vitae unde quidem eius.	1989
PostgreSQL	BSD	1989
MySQL	Komercyjna/GPL	1995
MariaDb	GPL	2009

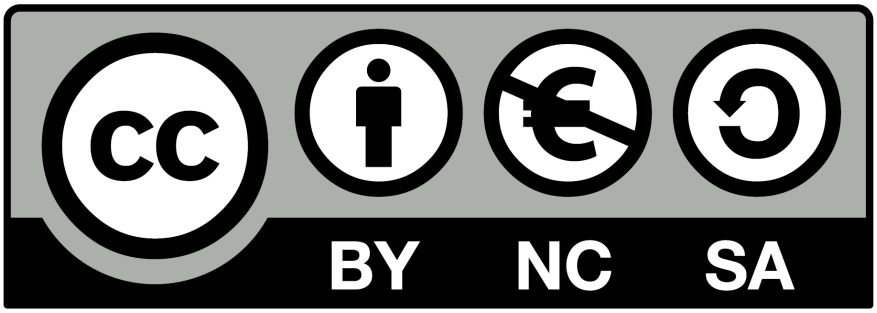
Wymuszone przełamanie strony.

2.3. Mój patent na obrazki

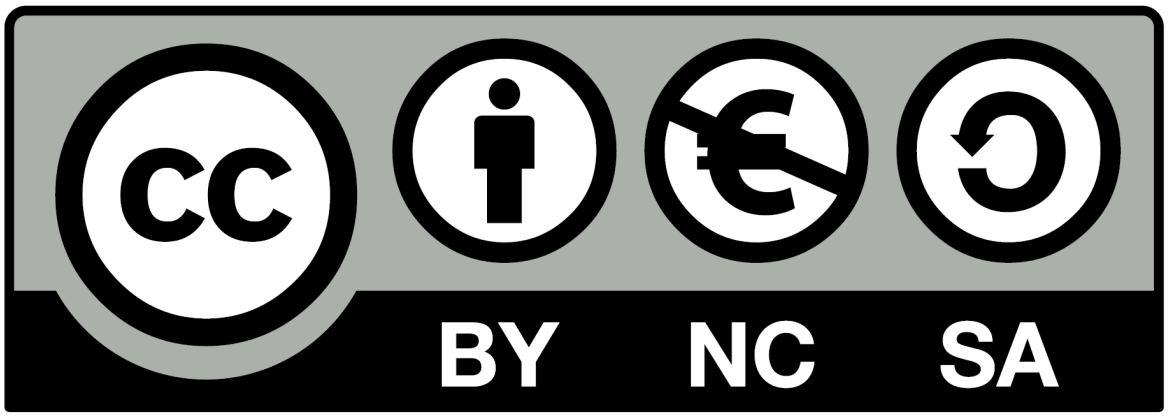
Zerknij w kod, narzuciłem trzy wymiary na zdjęcia ograniczając wysokość i szerokość - w praktyce bardzo przydatne i wygląda estetycznie. Odnośnik do zdjęcia 2.3.



Rys. 2.1: Mały obrazek



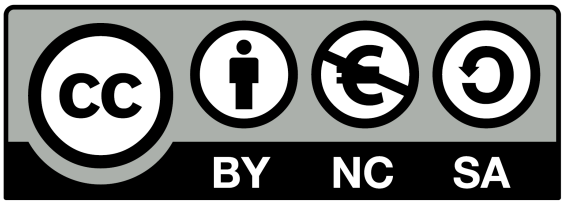
Rys. 2.2: Średni obrazek



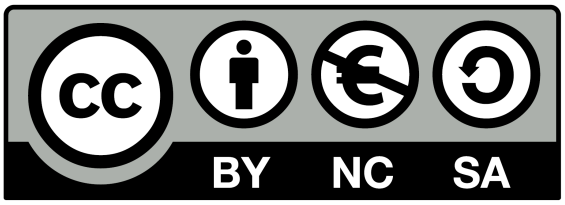
Rys. 2.3: Duży obrazek

Bo czasami potrzeba w dwóch kolumnach: 2.4 A czasamu potrzeba tylko dwa obrazy na jedenj stronie 2.6

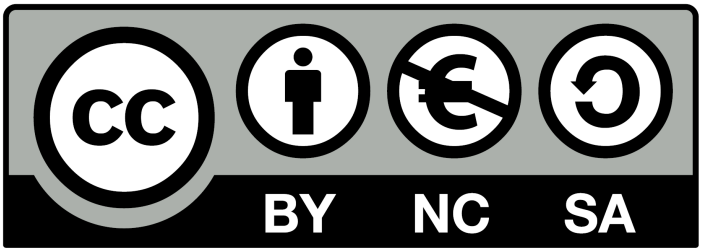
Wymuszone przełamanie strony. Raczej nie polecam do wykorzystania w praktyce, przyspaża zbędnej roboty. Ale zakomentuj to przełamanie i zobacz co zrobi z obrazkami!



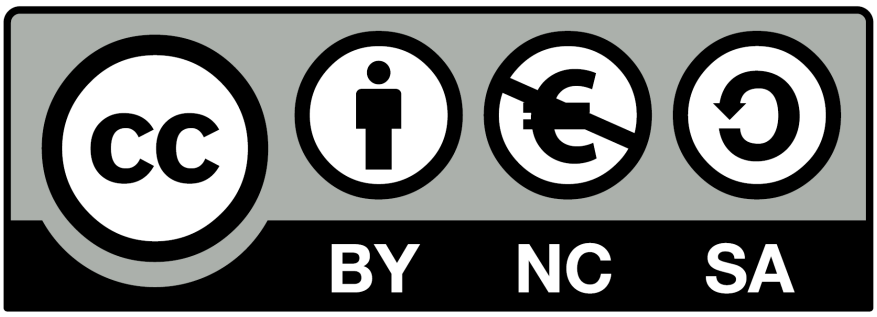
Rys. 2.4: pierwsza kolumna



Rys. 2.5: Druga kolumna



Rys. 2.6: Qui placeat et deserunt consequatur eos sed.



Rys. 2.7: Quis voluptatem rem et consequatur.

2.4. Kod - to co najciekawsze

Środowisko `onepage` jest nadpisaniem `mini page`, tak aby listingi nie były dzielone na strony, jak chcesz podzielić: wywal `onepage`. A przy okazji zobacz jak w `lcode` formatuję kod w linii, `int *ptr = &number; //:3`.

Listing 2.1: Przykład C#

```
1  namespace Example
2  {
3      public class Singleton<T> where T : class, new()
4      {
5          private static T instance;
6
7          public static Instance
8          {
9              get
10             {
11                 if (instance == null)
12                     instance = new();
13                 return instance;
14             }
15         }
16     }
17 }
```

A teraz `NAJLEPSZE!`, odwołanie do linii w kodzie. Na listingu 2.1 w linii 3 jest takie fajne słówko `where`. Wymagane jest dodanie np. `escapechar=‘` do argumentów `lstlisting`.

Listing 2.2: Przykład bash (i jakiś teskt et numquam omnis illo a tempora)

```
1  #!/bash/sh
2  version=$(git describe --tags $(git rev-list --tags --max-count=1))
3  echo "$version"
4
5  path="latex-build/main.pdf"
6  mkdir -p versions
7
8  cp "$path" "versions/PracaDyplomowa-$version.pdf"
9  cp "$path" "versions/PracaDyplomowa-LAST.pdf"
10
11 cd versions
12 start .
13 cd ..
```

Rozdział 3

Użyte narzędzia i technologie

3.1. Język Python

Język **Python** jest bardzo popularnym, nowoczesnym oraz wysokopoziomowym językiem programowania. Czytając artykuły i inne treści na temat historii **Python** [13] [12] dowiadujemy się, że język powstał w 1991 roku. Stworzony przez Guido van Rossum podczas swojej pracy w laboratorium w Centrum Matematyki i Informatyki w Amsterdamie, pierwotnie tworzony był z myślą zastąpienia rozwijanego w latach osiemdziesiątych języka **ABC**. Samą nazwę język Python nazwę zawdzięcza popularnemu serialowi komediowemu emitowanego przez BBC w latach siedemdziesiątych - "Latający Cyrk Monty Pythona", którego Guido był fanem. Projekt początkowo zakładał stworzenie prostego w użyciu, zwięzłego i wysokopoziomowego języka, głównie na potrzeby pracy w Amsterdamskim laboratorium. Z biegiem czasu **Python** stał się rozwijanym przez społeczność projektem **Open Source**, nad którym czuwała organizacja non-profit założona przez Guido von Rossum'a, **Python Software Foundation**.

Python jest multi-paradygmatowym językiem, gdzie programowanie obiektowe i strukturalne są w pełni wspierane, wiele cech języka wspiera programowanie funkcyjne i aspektowe. Wiele innych paradygmatów jest wspieranych dzięki modularnym rozszerzeniom do języka. **Python** w przeciwieństwie do języków statycznie typowanych takich jak **C**, **Rust** czy **TypeScript**, stosuje typowanie dynamiczne, które sprawdza poprawność i bezpieczeństwo typów w programie, dynamicznie, podczas jego egzekucji, w przeciwieństwie do typowania statycznego, gdzie typy sprawdzane są podczas kompilacji kodu. Dodatkowo **Python** posiada kombinacje zliczania referencji oraz cyklicznego **Garbage Collector**'a. Architektura języka **Python** oferuje wsparcie, dla programowanie funkcyjnego zgodnego z tradycją języka **List**. Język posiada typowe, dla języków funkcyjnych funkcje takie jak `filter`, `map`, `reduce`, `lambda`, list comprehensions, słowniki oraz generatory. Standardowa biblioteka języka posiada moduły `itertools` oraz `functools`, które implementują narzędzia funkcjonalne zapożyczone z języków **Haskell** oraz **Standard ML**.

Zamiast polegać na wbudowanej funkcjonalności w jądro języka, **Python** został zaprojektowany tak, aby być najmożliwiej elastyczny, aby mógł współdziałać z różnymi odrębnymi modułami. Sama kompaktowa modularność sprawiła, że język stał się popularnym sposobem, na dodawanie programowalnego interfejsu to istniejących już aplikacji oraz języków programowania. **CPython** jest referencyjną implementacją **Python**'a, napisaną w **C**, która spełnia standard **C89** z niektórymi cechami standardu **C99**. Jedną z charakterystyk

implementacji **CPython**'a jest to, że zespół odpowiedzialny za rozwój oraz utrzymanie kodu jego interpretera preferuje odrzucanie poprawek do kodu mających na celu marginalną poprawę szybkości i sprawności interpretera w zamian za zachowanie czystości i czytelności kodu źródłowego. Taka długoterminowa postawa umożliwiła powstanie powstanie innych implementacji języka **Python** opartych o inne rozwiązania, bądź inne języki za pomocą których napisany został sam interpreter. Dzięki różnorodności implementacji **Python**'a, jego modularności i łatwej rozbudowie o obce rozszerzenia programista może zdecydować się na przeniesienie wrażliwych na czas wykonywania funkcjonalności do odrębnych modułów napisanych w innych językach takich jak **C**, lub **Rust**, albo użyć wyspecjalizowanej do tego odmiany samego interpretera takiej jak **PyPy**, która posiada tak zwany **JIT**, czyli **Just-In-Time compiler**, pozwalający na optymalizację kodu na system bądź architekturę docelową danego programu.

Dzięki tak rozbudowanemu środowisku oraz społeczności otaczającej język powstała ogromna biblioteka paczek, dla języka **Python**, która często podawana za jedną z największych zalet języka. W ten sposób **Python** stał się swoistym odpowiednikiem *Lingua Franca* wśród programistów. Przez co często dziedziny zajmujące się całkowicie różnymi dziedzinami takimi jak Web Development, Automatyzacja, Bazy Danych, Aplikacje Mobilne, Testowanie Oprogramowania, Analiza Danych oraz Uczenie Maszynowe.

3.2. Biblioteka PyTorch

PyTorch [9] [6] jest Open Source'ową biblioteką służącą do uczenia maszynowego. Jest bazowana na bibliotece **Torch** napisanej w języku **Lua**. Z powodu niszowości języka **Lua**, oraz braku modularności i możliwości rozbudowywania o nowe funkcjonalności za pomocą zewnętrznych modułów i paczek, powstał **PyTorch**, czyli, biblioteka **Torch**, ale zaimplementowana w języku **Python**. Dzięki temu **PyTorch** może korzystać z bardzo rozbudowanego środowiska **Python**, które oferuje dużą ilość naukowych paczek, między innymi takich jak **NumPy**.

Jedną z największych zalet biblioteki **PyTorch** jest możliwość programowanie imperatywnego. Jest to przeciwieństwem do bibliotek takich jak **TensorFlow** i **Keras**, które ze względu na poleganie głównie na językach takich jak **C** i **C++**, oferują jedynie możliwość programowania symbolicznego. Większość **Python**'owego kodu jest imperatywne jako, że jest to dynamicznie interpretowany język. W sytuacji symboliczne zachodzi przeciwieństwo, ponieważ zachodzi bardzo wyraźne rozróżnienie pomiędzy zdefiniowaniem grafu komputacyjnego, a jego kompilacją. W przypadku imperatywnym komputacja zachodzi w momencie jej wywołania, nie we wcześniej zoptymalizowanym punkcie w kodzie. Podejście symboliczne pozwala na większą optymalizację, a imperatywne takie jak **PyTorch** pozwalają na większą swobodność, oraz używanie natywnych cech, funkcjonalności i rozszerzających modułów języka **Python**. Drugą największą zaletą biblioteki **PyTorch** są dynamiczne grafy komputacyjne, które w przeciwieństwie do bibliotek takich jak **TensorFlow** generują je statycznie przed uruchomieniem programu. **PyTorch** umie generować i modyfikować je dynamicznie podczas działania programu.

3.3. Biblioteka NumPy

NumPy [10] [8] [3] [2] jest Open Source biblioteką stworzoną, dla języka programowania **Python**, dodaje wsparcie, dla dużych wielowymiarowych tablic i macierzy wraz z dużą kolekcją wysokopoziomowych funkcji matematycznych, które pozwalają operować na wspomnianych macierzach. Poprzednikiem biblioteki **NumPy** był, **Numeric**, oryginalnie stworzony przez Jim'a Hugunin'a wraz z kontrybucjami kilku innych developerów. W roku 2005, Travis Oliphant stworzył projekt **NumPy** włączając w to właściwości oraz funkcjonalności **Numeric**'a. **Python** nie był oryginalnie stworzony do numerycznej komputacji, ale już we wczesnym życiu języka różne towarzystwa naukowe i inżynieryjne wyrażały swoje zainteresowanie językiem. **NumPy** adresuje problem powolności języka **Python** poprzez zapewnienie wielowymiarowych macierzy, funkcji i operacji, które są wydajne obliczeniowo operując na macierzach.

Używanie biblioteki **NumPy** w **Python**'ie funkcjonalnością przypomina programowanie w środowisku **MATLAB**, jako, że oba są interpretowane, mają podobną składnie, oba pozwalają użytkownikowi pisać szybkie i wydaje programy tak długo jak operacje przeprowadzane są na macierzach. W przeciwieństwie do **MATLAB**'a, **NumPy** nie oferuje tak wielkiej ilości dodatkowych narzędzi. Oferuje odwrotne podejście, gdzie to inne paczki/narzędzia korzystają u swoich podstaw z biblioteki **NumPy**. Dzięki zaawansowanej integracji z **Python**'em oraz byciu podłożem, dla zasadniczej większości paczek i narzędzi służących celom obliczeniowo naukowym takim jak **SciPy**, **SymPy**, **Scikit-Learn** i wielu innym **NumPy** stał się podstawową warstwą, dla takich narzędzi a tym samym umożliwia im prostą komunikację między sobą jako, że macierze na których operują są macierzami biblioteki **NumPy**.

Główną funkcjonalnością biblioteki **NumPy** jest jej `ndarray`, struktura danych, reprezentująca *n*-wymiarową macierz. Wewnętrzna niskopoziomowa implementacja owych macierzy polega na kroczących widokach w pamięci. Takowe dane muszą być homogenicznego typu. Takie widoki pamięci mogą być również bufferami zaalokowanymi z poziomu innych języków takich jak **C**, **C++**, czy **Fortran**. Powoduje to ogromną optymalizację, jako, że eliminuje to potrzebę kopiowania i przenoszenia danych.

3.4. Biblioteka Matplotlib

Biblioteka **Matplotlib** [7] [11] [10] [1] jest najpopularniejszym **Python**'owym narzędziem służącym do tworzenia wykresów, grafów, histogramów, obrazów, wielowymiarowych grafów i rysunków służących do wizualizacji danych. Oryginalnie stworzona prze z John D. Hunter'a miała za zadanie tworzyć przyzwoicie wyglądające wykresy, które można by poddać publikacji. Mimo, że inne biblioteki są dostępne większość programistów używa **Matplotlib**'a jako, że jest on najpopularniejszy oraz najbardziej i najlepiej rozbudowany ze wszystkich nie wspominając o jego wpasowaniu w istniejący ekosystem Data Science.

Matplotlib zapewnia obiektowo zorientowane **API** do tworzenia i używania generowanych wykresów w toolkitach **GUI** takich jak **Tkinter wxPython**, **Qt**, lub **GTK**. Dodatkowo istnieje również proceduralny interfejs "**pylab**" bazujący na maszynie stanu (podobnie do **OpenGL**), zaprojektowany, aby przypominać interface **MATLAB**'a.

Pyplot to moduł **Matplotlib**'a, który jest zaprojektowany z myślą bycia używalnym pod kątem programistycznym tak jak **MATLAB**, ale z dodatkową zaletą pozostawania w przestrzeni języka **Python**, która jest Open Source i darmowa.

3.5. PyGame

PyGame [5] [4] jest cross-platform'ową biblioteką stworzoną, dla języka **Python** zaprojektowaną z myślą tworzenia prostych gier i animacji. Zawiera w sobie biblioteki odpowiadające za grafikę i dźwięk. **PyGame**, był oryginalnie napisany przez Pete Shinners'a, w celu zastąpienia **PySDL** po tym jak jego developemnt został zatrzymany. Od roku 2000 jest to projekt społeczności i został objęty licencją **GNU Lesser General Public License**, która pozwala na dystrybucje **PyGame**'a zarówno wraz z Open Source'owym oprogramowaniem jak i tym prawnie zastrzeżonym prawami autorskimi.

Rozdział 4

Sieć neuronowa

4.1. Sieć neuronowa

Ukryty link do spisu treści Kliknij na numer strony, a pod nim masz ukryty odnośnik do spisu treści. Bardzo ułatwia skakanie po pdf'ie.

Używasz vscode? Zobacz na polecane wtyczki w README.

Jak wydzilić z tesktu Jak cokolwiek chcesz oddzielic z tekstu możesz użyć `onepage[Xmm]`.

Alias dolor odit. Velit et ut harum. Quos ullam enim suscipit qui omnis dolorum.

Reszta tekstu. Doloremque voluptas sit mollitia eos ut aut. Qui et distinctio vitae. Possimus et in. Est provident qui sequi est nobis cupiditate magni. Recusandae animi aut non ea autem ipsam dolores hic repellendus.

I chyba nic więcej nie potrzeba. Powodzenia!

Rozdział 5

Paczka Python

5.1. Paczka Python

Ukryty link do spisu treści Kliknij na numer strony, a pod nim masz ukryty odnośnik do spisu treści. Bardzo ułatwia skakanie po pdf'ie.

Używasz vscode? Zobacz na polecane wtyczki w README.

Jak wydzilić z tesktu Jak cokolwiek chcesz oddzielic z tekstu możesz użyć `onepage[Xmm]`.

Alias dolor odit. Velit et ut harum. Quos ullam enim suscipit qui omnis dolorum.

Reszta tekstu. Doloremque voluptas sit mollitia eos ut aut. Qui et distinctio vitae. Possimus et in. Est provident qui sequi est nobis cupiditate magni. Recusandae animi aut non ea autem ipsam dolores hic repellendus.

I chyba nic więcej nie potrzeba. Powodzenia!

Rozdział 6

Użycie

6.1. Użycie

6.1.1. Własna gra - Snake

Ukryty link do spisu treści Kliknij na numer strony, a pod nim masz ukryty odnośnik do spisu treści. Bardzo ułatwia skakanie po pdf'ie.

Używasz vscode? Zobacz na polecane wtyczki w README.

Jak wydzilić z tesktu Jak cokolwiek chcesz oddzielic z tekstu możesz użyć `onepage[Xmm]`.

Alias dolor odit. Velit et ut harum. Quos ullam enim suscipit qui omnis dolorum.

6.1.2. Cudza gra - PyGame Examples

Reszta tekstu. Doloremque voluptas sit mollitia eos ut aut. Qui et distinctio vitae. Possimus et in. Est provident qui sequi est nobis cupiditate magni. Recusandae animi aut non ea autem ipsam dolores hic repellendus.

I chyba nic więcej nie potrzeba. Powodzenia!

Literatura

[1] Matplotlib. https://en.wikipedia.org/wiki/Matplotlib, 2022.

[2] Numpy. https://en.wikipedia.org/wiki/NumPy, 2022.

[3] Numpy manual. https://numpy.org/doc/stable/, 2022.

[4] Pygame. https://en.wikipedia.org/wiki/Pygame, 2022.

[5] Pygame documentation. https://www.pygame.org/wiki/GettingStarted, 2022.

[6] Pytorch. https://pytorch.org/docs/stable/index.html, 2022.

[7] Users guide. https://matplotlib.org/stable/users/index, 2022.

[8] E. Bressert. *SciPy and NumPy*. O'Reilly Media, Inc., 2012.

[9] E. S. Luca Pietro Giovanni Antiga, Thomas Viehmann. *Deep Learning with PyTorch*. Manning Publications, 2020.

[10] W. McKinney. *Python for Data Analysis, 3rd Edition*. O'Reilly Media, Inc., 2021.

[11] S. R. Poladi. *Matplotlib 3.0 Cookbook*. Packt Publishing, 2018.

[12] Wikipedia. Python (programming language). https://en.wikipedia.org/wiki/Python_(programming_language).

[13] A. Zamczała. Python ma już 30 lat! od czego wszystko się zaczęło? 2021.

Spis rysunków

Spis tabel

Indeks rzeczowy