

Uniwersytet WSB Merito

Uczenie Maszynowe

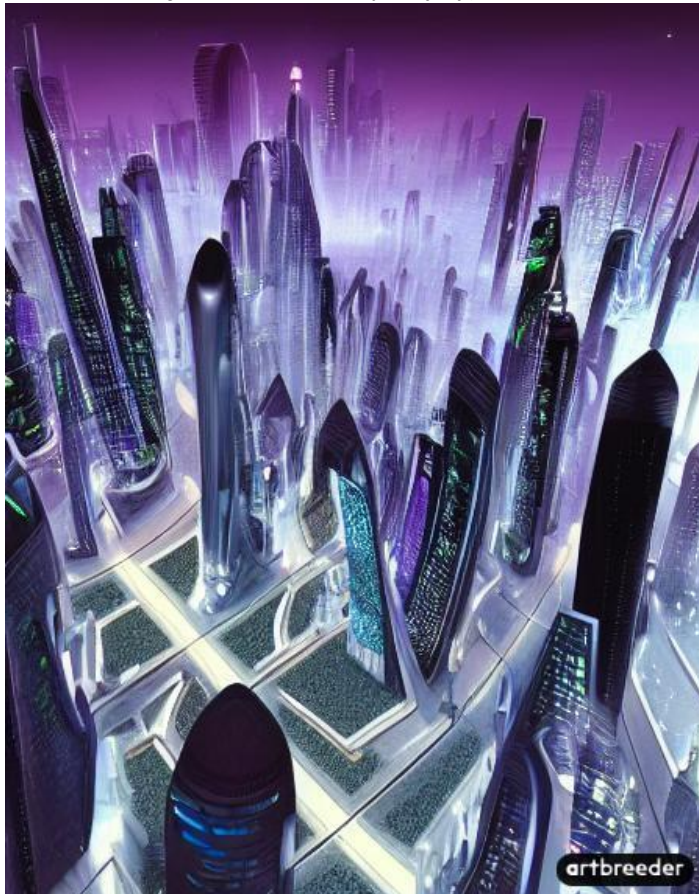
Ćwiczenia 7 - zadania

1. Mateusz Stania i Dawid Przelazły
2. m.stania@outlook.com
3. Numer albumu. 134789
4. 28.05.2023
5. Link do repo na Github https://github.com/zabato16/UM_CW_7

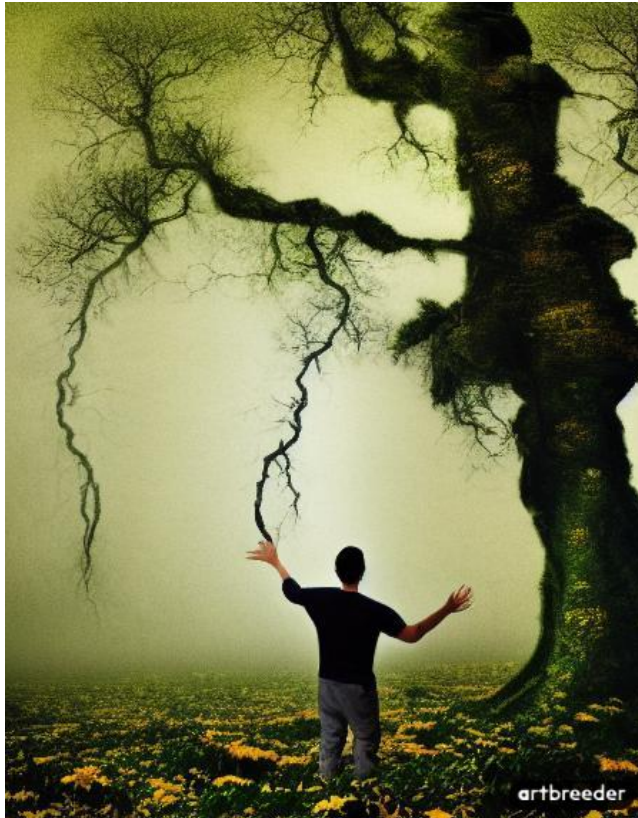
Rozdział 1 – Sieci Generatywne GAN

Zadanie 1 - ArtBreeder.com

Dla strony <https://artbreeder.com> wykonać następujące stworzyć konto i używając narzędzia collage spróbować wygenerować po jednym obrazie dla 10 wybranych kategorii z poniższych tematyk (dozwolone są też inne własne pomysły):



Rysunek 1 Futurystyczne miasto



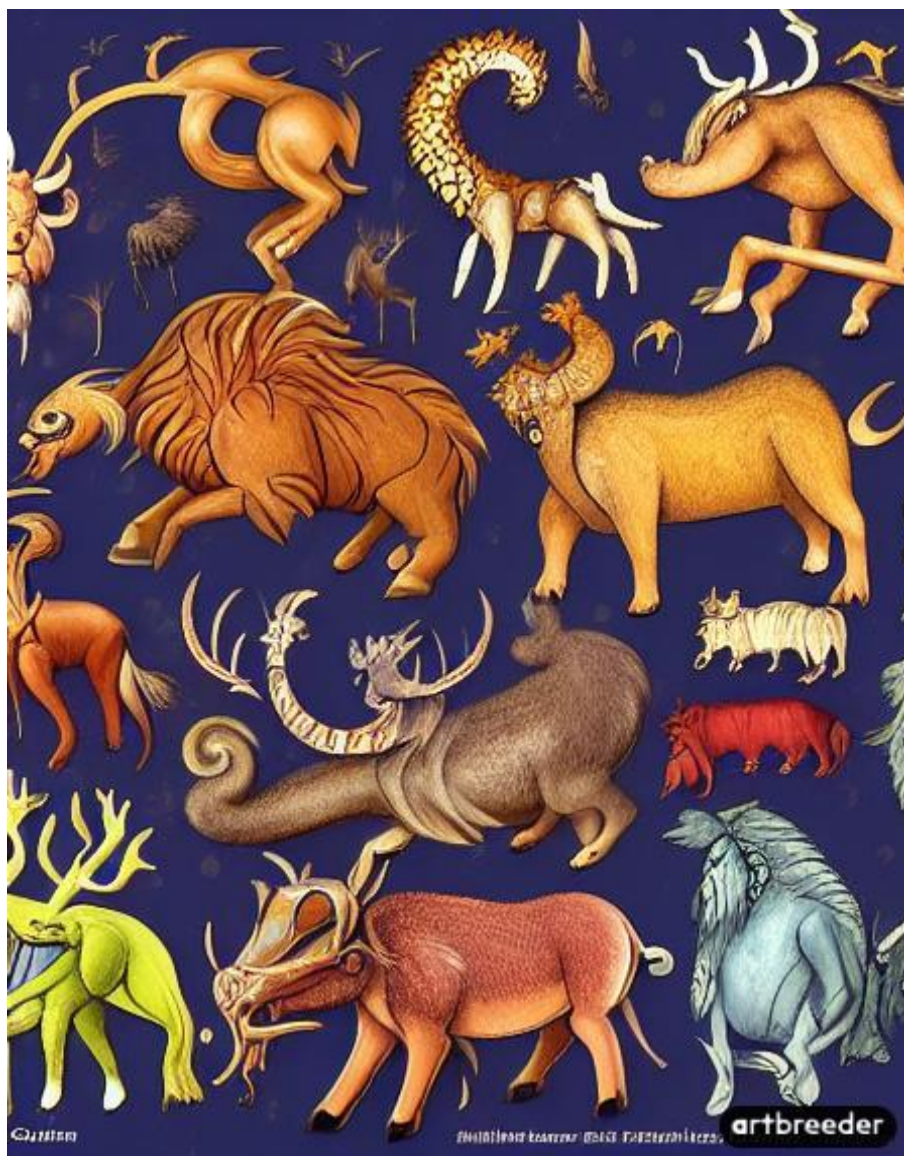
Rysunek 2 Magiczny las



Rysunek 3 Taniec i ruch w abstrakcyjnej formie



Rysunek 4 Tajemnicza wyspa



Rysunek 5 Magiczne stworzenia i stwory



Rysunek 6podwodny świat



Rysunek 7kot w kosmosie



Rysunek 8przygoda w kosmosie



Rysunek 9 Taniec i ruch w abstrakcyjnej formie



Rysunek 10 Futurystyczne miasto

Zadanie 2 - RunWayML.com

Strona <https://runwayml.com> pozwala na tworzenie własnych projektów graficznych, dźwiękowych. Założenie darmowego konta pozwala na wygenerowanie co najmniej 105 obrazów.

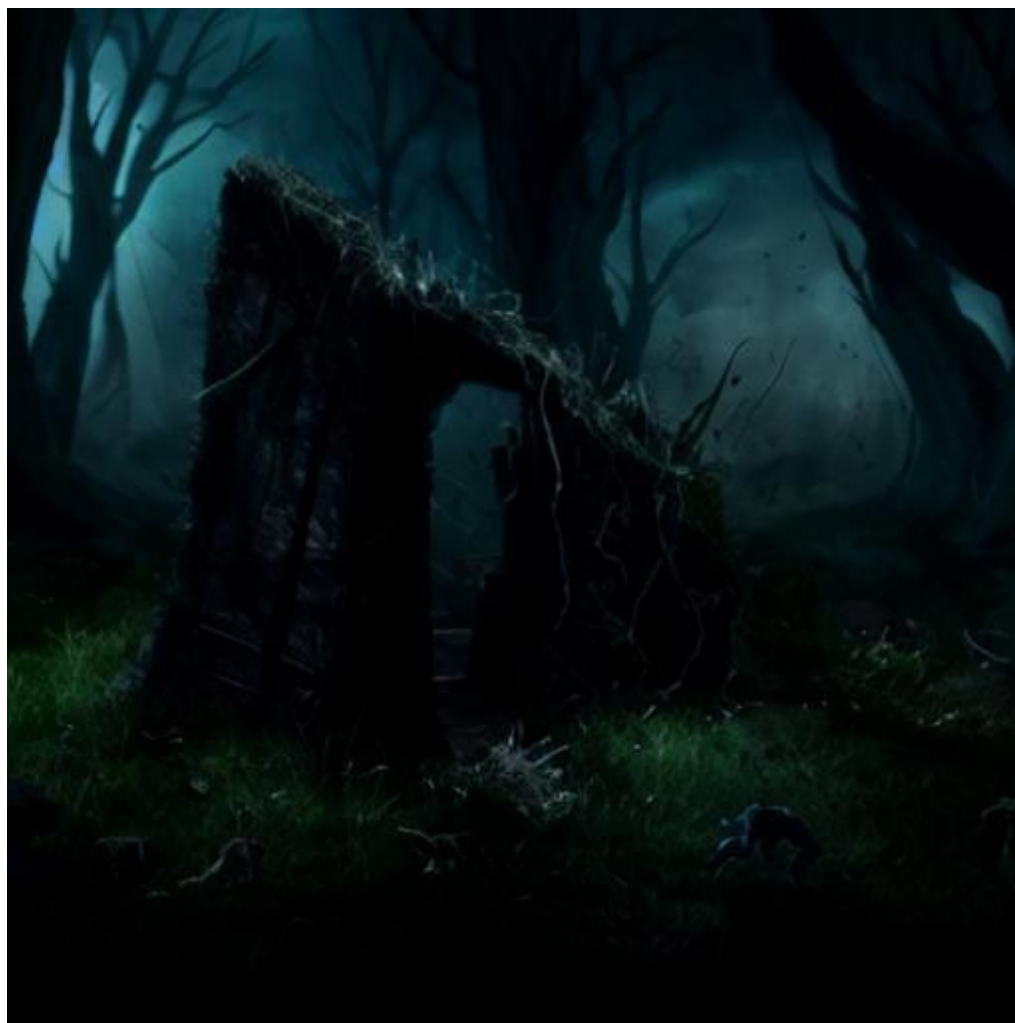
Oczywiście konto założone.

Zadanie 3 - Proste obrazy

Dla tej samej strony wygenerować 10 obrazów z dowolnej kategorii z poniższych (można wymyślić własne kategorie):



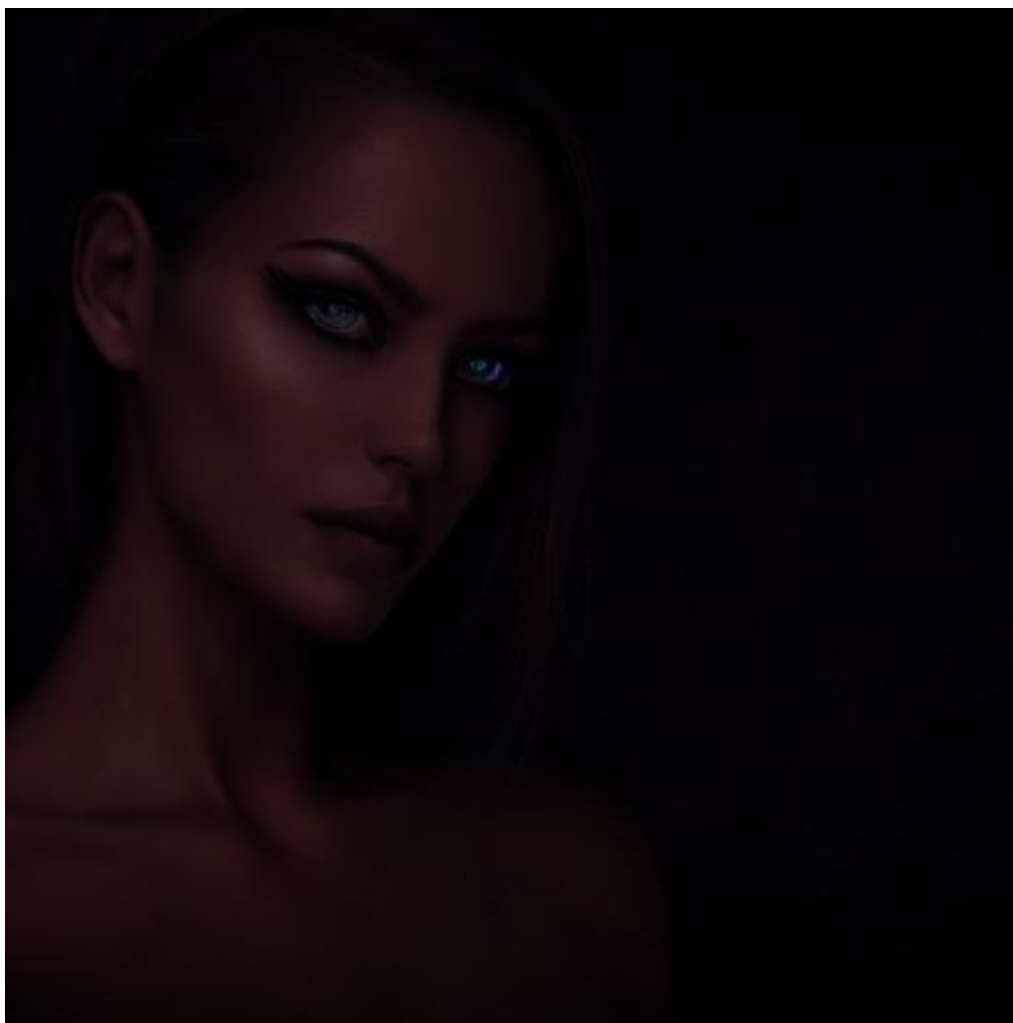
Rysunek 11 Krajobraz



Rysunek 12 Mroczny las



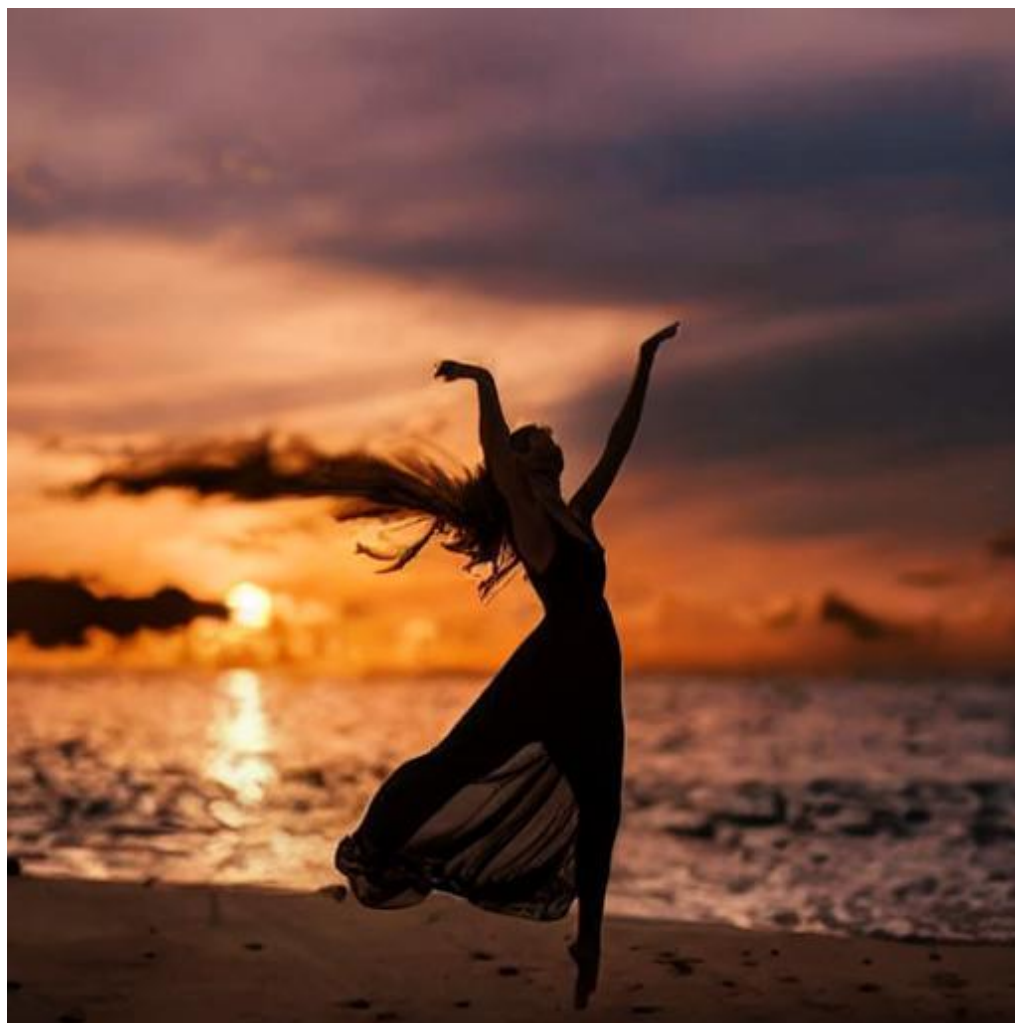
Rysunek 13 Las z horroru



*Rysunek 14*Kobieta w ciemności



Rysunek 15 Zwierze z horroru



Rysunek 16 Taniec o zachodzie



Rysunek 17 Zwierze z horroru



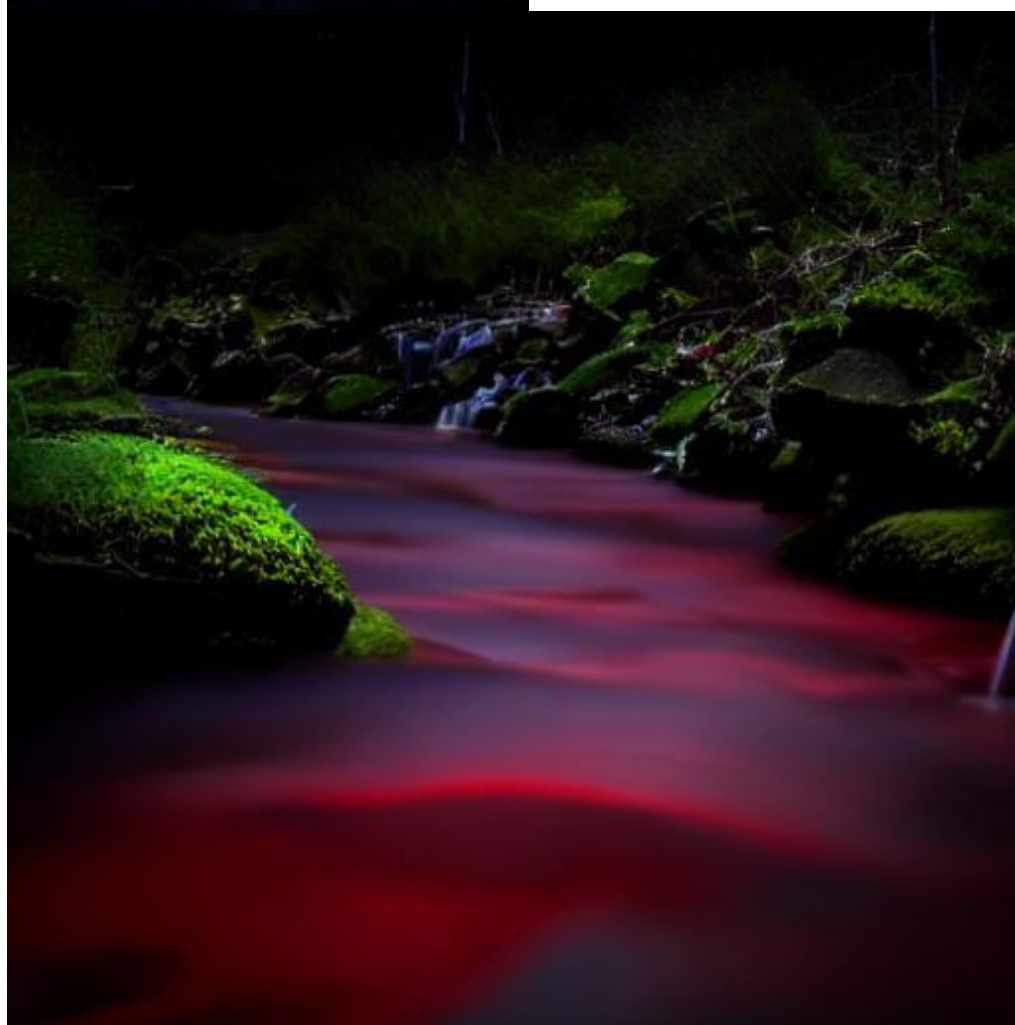
Rysunek 18 Postać z horroru

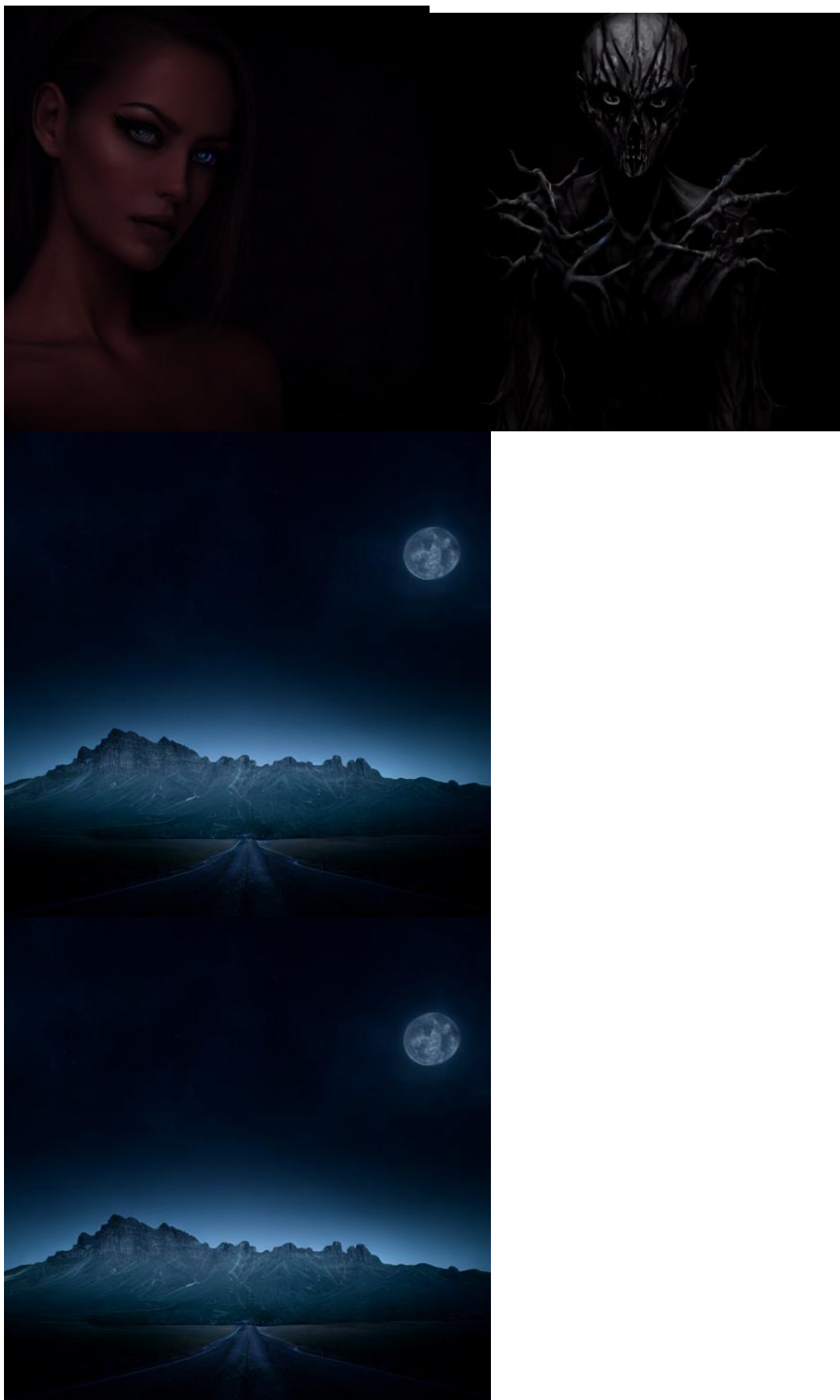


Rysunek 19 Futurystyczne stworzenie



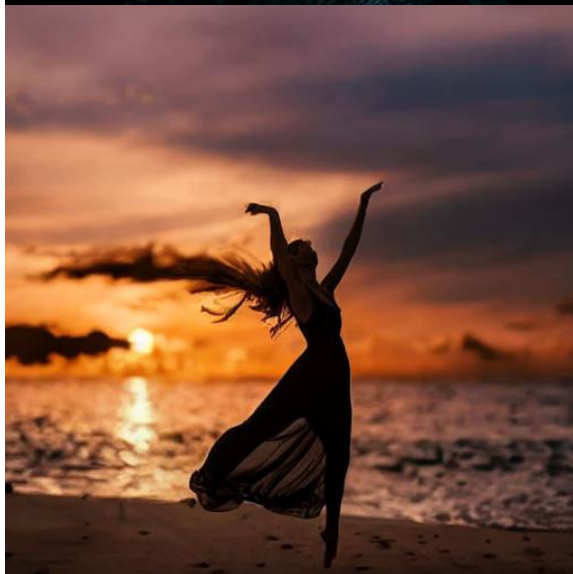
Rysunek 20 Nocny krajobraz













Rysunek 21 Tęczowy jednorozec



Zadanie 4 - Zaawansowane generowanie

Użyć <https://chat.openai.com> do wygenerowania bardziej zaawansowanych promptów do tworzenia obrazów.

Wygenerować w ten sposób znowu 10 obrazów podobnie jak w 1.2.1

1. Wygeneruj prompt do stworzenia obrazu na temat tęczowego jednorożca pijącego wodę z brzegu krwistej rzeki

ChatGPT

Tytuł: "Tęczowy jednorożec nad krwistą rzeką"

Opis:

Przedstaw obraz tęczowego jednorożca, który stoi na brzegu krwistej rzeki i pije wodę z jej nurtu. Jednorożec powinien być w pełni ukazany, z eleganckim, długim rogiem na czole, jedwabistym włosiem i puszystym ogonem. Jego sylwetka powinna być wyraźnie oświetlona przez słońce, które przebija się przez chmury. Tęczowe kolory powinny emanować z jego grzywy, ogona i rogów.



2 User
a teraz krwista rzeka

Krwista rzeka powinna mieć głęboki, czerwony odcień, przypominający krew. Jej woda powinna być płynna, a fale delikatnie przepływać wokół jednoróżca. Woda może być odbijana przez skały lub kamienie na brzegu rzeki, dodając dramatyzmu i kontrastu do obrazu.

W tle można przedstawić leśne otoczenie z bujną roślinnością, która wyrasta z ziemi wokół rzeki. Niebo powinno być niebieskie z delikatnymi chmurami, które tworzą atmosferę tajemniczości i magii.

Uwaga: Możesz dostosować i rozbudować ten opis według własnej wizji artystycznej.



Uczenie Maszynowe

3 User

a teraz dla obrazu tytuł "Kobieta w horrorowej scenarii"

ChatGPT

Tytuł: "Mroczne oblicze"

Opis:

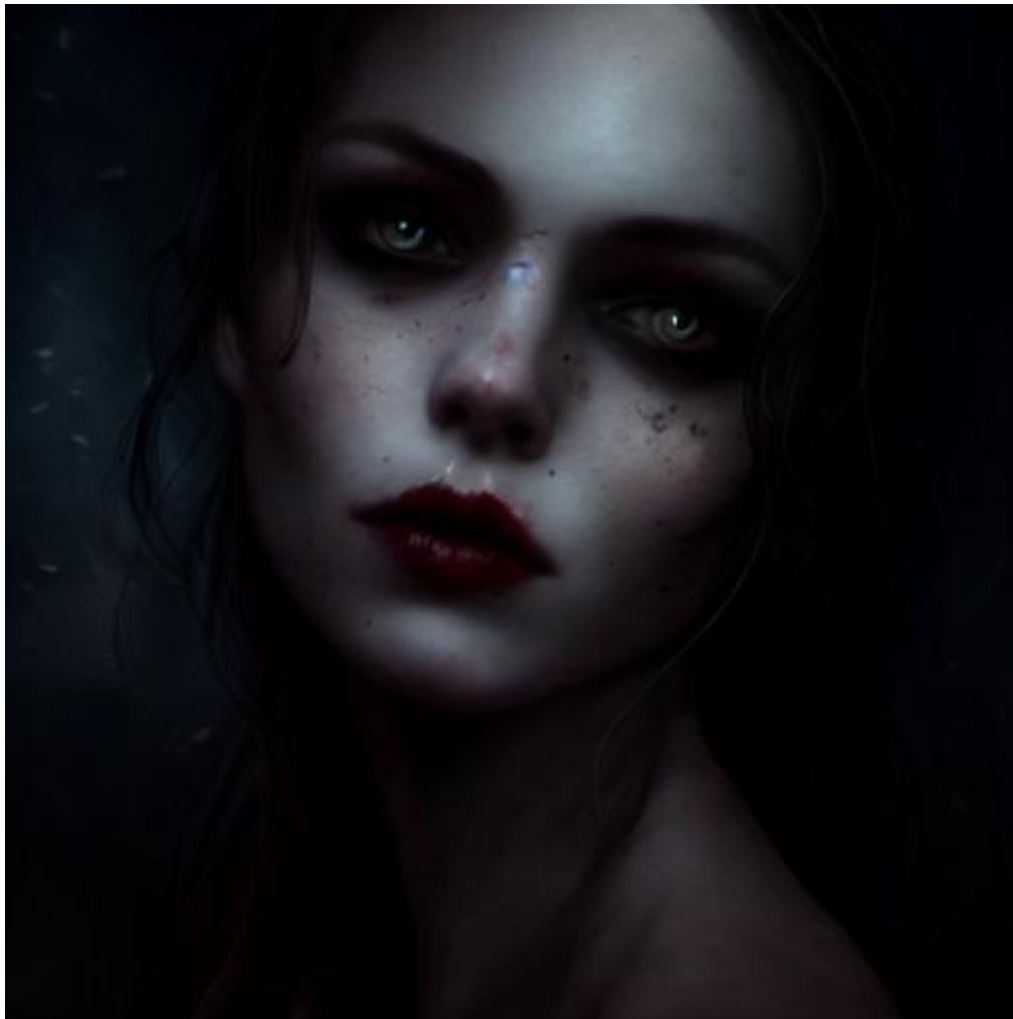
Przedstaw obraz kobiety w horrorowej scenarii, która wprowadza widza w atmosferę napięcia i niepokoju. Kobieta powinna być głównym punktem centralnym kompozycji, otoczona przez mroczne elementy.

Jej wygląd powinien być nieco tajemniczy i niepokojący. Może mieć bladą skórę, ciemne, wyostrome oczy lub delikatnie widoczne rysy przypominające upiorny makijaż. Jej wyraz twarzy powinien oddawać mieszaninę strachu, determinacji i tajemniczości.

Sceneria wokół niej powinna być pełna mrocznych detali. Możesz przedstawić opuszczony, zrujnowany budynek lub nawiedzony las w tle. Dodaj elementy, takie jak powiewające gałęzie, mgła lub zniszczone otoczenie, aby podkreślić atmosferę grozy.

Dodatkowe szczegóły, takie jak skradające się cienie lub tajemnicze postaci w oddali, mogą dodatkowo wzmocnić napięcie. Kolory mogą być głównie stonowane, z dominacją odcieni szarości, czerni i ciemnych tonów, ale możesz również wykorzystać kontrastujące akcenty, np. czerwienie lub purpury, aby podkreślić niebezpieczeństwo.

Uwaga: Podane wyżej szczegóły są jedynie sugestiami. Możesz dostosować opis obrazu według własnej wizji artystycznej i koncepcji horroru.



4User

sam wymyśl inny temat

ChatGPT

Tytuł: "Podróżnik kosmiczny w otchłaniach wszechświata"

Opis:

Przedstaw obraz podróżnika kosmicznego, który eksploruje głębokie otchłanie wszechświata. Jego postać powinna być centralnym punktem kompozycji, wyrażającą ducha odkrywczości i odwagi.

Podróżnik może być przedstawiony w specjalnym kombinezonie kosmicznym, który podkreśla jego postać i związaną z przestrzenią kosmiczną. Może mieć na sobie hełm, który odbija światło gwiazd, a na tle ciemności wszechświata.

Otchłanie wszechświata powinny być pełne gwiazd, galaktyk i mgławic, tworząc efekt wspaniałej, niewyobrażalnej przestrzeni. Możesz wykorzystać różne kolory, aby oddać piękno i tajemniczość kosmosu - od głębokiej czerni i błękitów aż po jasne odcienie różu i purpury.

Dodatkowe elementy, takie jak asteroidy lub kosmiczne pyłki, mogą dodać dynamiki i głębi obrazu. Możesz również pokazać inne statki kosmiczne w oddali, podkreślając skalę i potęgę wszechświata.

W tle obrazu można przedstawić dalekie galaktyki, które zdają się nie mieć końca. Możesz również uwypuklić efekt głębi, dodając mgliste halo lub światło, które emanuje z nieznanego źródła.

Uwaga: Powyższy opis jest sugestią. Możesz dostosować go i rozbudować według własnej wizji artystycznej i fascynacji kosmosem.



Uczenie Maszynowe

5 User
teraz piękny krajobraz z mnóstwem szczegółów
ChatGPT
Tytuł: "Harmonia Natury"

Opis:

Przedstaw obraz pięknego krajobrazu, w którym natura ukazuje się w pełnej okazałości. Pejzaż powinien być pełen szczegółów, które nadają mu bogactwo i autentyczność.

W centrum kompozycji można umieścić malownicze wzgórze lub pagórek, który wznosi się nad szeroką doliną. Zadbaj o różnorodność terenu, aby nadać krajobrazowi głębię i zróżnicowanie. Wzgórze może być pokryte soczystą, zieloną trawą lub kolorowymi kwiatami, które dodają mu życia i energii.

W tle można uwzględnić majestatyczne góry lub pasmo górskie, które sięga ku niebu. Ich szczyty mogą być pokryte śniegiem lub utopione w chmurach, podkreślając ich imponującą wysokość.

W dolinie lub na polach można umieścić różnorodne elementy, takie jak strumienie, jeziora lub rzeki. Możesz przedstawić przejrzystą, błękitną wodę, która odbija otaczającą przyrodę. Dodanie delikatnych fal lub spływających kaskad wody może nadać dynamiki i ruchu scenie.

Nie zapomnij o roślinności. Możesz umieścić drzewa w różnych kształtach i odcieniach zieleni. Dodaj kwitnące krzewy, dzikie kwiaty lub gęsty las, aby podkreślić bujność i różnorodność przyrody.

Dodatkowe detale, takie jak ptaki unoszące się w powietrzu, motyle tętniące życiem czy żółwie zanurzone w jeziorze, mogą wzbogacić obraz i wprowadzić uczucie obecności i harmonii z naturą.

Uwaga: Powyższy opis ma charakter sugestii. Możesz dostosować i rozbudować go zgodnie z własną wizją artystyczną, aby stworzyć unikalny i piękny krajobraz.



Uczenie Maszynowe

6 User

teraz dla pałac kultury i nauki na tle polskich gór

ChatGPT

Tytuł: "Symfonia architektury i natury"

Opis:

Przedstaw obraz Pałacu Kultury i Nauki w Warszawie, majestatycznie wznoszącego się na tle polskich gór. Kontrast między monumentalną architekturą pałacu a pięknem naturalnego krajobrazu dodaje obrazowi unikalnego uroku.

Pałac Kultury i Nauki powinien być ukazany w całej swojej okazałości, z dominującą wieżą, detalami architektonicznymi i charakterystycznymi konturami. Jego stalowa struktura powinna błyszczeć w promieniach słońca, a refleksy światła mogą dodawać mu blasku i dynamiki.

W tle, jako kontrast dla budowli, przedstaw polskie góry, takie jak Tatry, Sudety lub Bieszczady. Możesz ukazać wznoszące się szczyty, pokryte śniegiem lub okryte lasami, tworzące majestatyczną scenę. Góry powinny emanować siłą i spokojem, symbolizując ich długowieczność i stabilność wobec przemijającej architektury.

Możesz również uwzględnić elementy przyrody, takie jak strumienie, jeziora lub lasy, które otaczają pałac i wprowadzają harmonię między budowlą a naturą. Ptaki unoszące się w powietrzu lub zwierzęta, takie jak jelenie czy rysie, mogą dodatkowo ożywić obraz i podkreślić obecność dzikiej przyrody w otoczeniu.

Ważne jest również zrównoważenie kompozycji, aby zarówno pałac, jak i góry, były równie istotne w obrazie. Możesz zastosować różne techniki malarskie lub światłocieniowe, aby podkreślić głębię i perspektywę, a także wyeksponować detale zarówno pałacu, jak i natury.

Uwaga: Powyższy opis jest sugestią. Możesz dostosować i rozbudować go według własnej wizji artystycznej, aby stworzyć unikalny obraz, który harmonijnie łączy architekturę z pięknem polskich gór.



7 User
zaskocz mnie następnym
ChatGPT
Tytuł: "Podróż przez czas i przestrzeń"

Opis:

Przedstaw abstrakcyjny obraz, który zaprasza widza do niezwyklej podróży przez czas i przestrzeń. Wyobraźnia jest kluczem do tego dzieła, więc pozwól sobie na eksperymenty i nietypowe połączenia.

W centralnym punkcie kompozycji przedstaw postać podróżnika, która emanuje tajemniczością i odkrywczością. Może to być postać ludzka w futurystycznym stroju lub abstrakcyjna forma, która symbolizuje ludzką duszę w podróży przez kosmos.

Wokół podróżnika układaj fragmenty krajobrazów, miast, galaktyk i innych elementów, tworząc kolaże, które przenoszą widza w różne epoki i miejsca. Możesz łączyć architekturę z przeszłości, teraźniejszości i przyszłości, tworząc unikalne zestawienia i perspektywy.

Kolory powinny być żywe i kontrastowe, aby podkreślić dynamikę podróży. Możesz użyć intensywnych odcieni niebieskiego, fioletu, żółci i pomarańczy, które symbolizują nieznaną sferę kosmosu i czasu.

Dodaj elementy poruszające się lub zakrzywiające przestrzeń, takie jak wiry, wirujące galaktyki lub rozciągające się struktury geometryczne. Dzięki nim obraz nabierze ruchu i głębi, a także wprowadzi uczucie podróży przez nieodkryte sfery wszechświata.

Pozwól swojej wyobraźni na swobodę i eksperymenty artystyczne, aby stworzyć niepowtarzalny obraz, który zaskoczy i zainspiruje widza.

Uwaga: Powyższy opis ma charakter sugestii. Możesz swobodnie dostosować i rozwijać go według własnej wizji artystycznej, aby stworzyć niesamowitą podróż przez czas i przestrzeń.



Uczenie Maszynowe

8 teraz, dziecko z transparentem w stylu anime

ChatGPT

Tytuł: "Młoda moc głosu"

Opis:

Przedstaw obraz uroczego dziecka w stylu anime, trzymającego przezroczyste tęczbowe transparentne. Wyraźnie ukazane cechy anime, takie jak duże, wyraźne oczy i kształtne, odznaczające się linie, pomogą w stworzeniu ujmującej postaci.

Dziecko powinno być ubrane w kolorową, radosną odzież, która podkreśli jego niewinność i energię. Może mieć uśmiechniętą buzię i delikatnie uniesione ręce, pokazując entuzjazm i pewność siebie.

Przezroczyste tęczbowe transparentne, które trzyma dziecko, może zawierać inspirujący napis, slogan lub wyraz przekazu. Może to być coś związanego z empatią, miłością, marzeniami czy odwagą. Napis na transparentnym powinien być czytelny, ale delikatny, pasujący do lekkości i uroku dziecka.

Tło obrazu można umieścić w pastelowych kolorach lub w stylizowanych, abstrakcyjnych motywach, które podkreślą atmosferę anime. Możesz również dodać subtelną ilustrację, która harmonizuje z przesłaniem transparentnego, na przykład kwiaty, motyle lub gwiazdy, symbolizujące nadzieję i piękno.

Dodatkowo, otaczające dziecko światło, cienie lub promienie mogą dodatkowo podkreślić jego obecność i wzmocnić aurę pozytywnej energii.

Uwaga: Powyższy opis jest sugestią. Możesz dostosować i rozbudować go według własnej wizji artystycznej, aby stworzyć ujmujący obraz anime przedstawiający dziecko z transparentem, które inspirowe swoją niewinnością i młoda mocą głosu.



Uczenie Maszynowe

9 Wygeneruj prompt do stworzenia obrazu na temat budowla na budowli

ChatGPT

Tytuł: "Miasto na wieży"

Opis:

Przedstaw obraz budowli, która sama w sobie stanowi fundament i podstawę dla kolejnej, mniejszej budowli. Wizualizacja ta może być oparta na koncepcji "budowla na budowli", gdzie architektura jest umieszczona warstwowo, tworząc unikalną strukturę.

Na pierwszym planie możesz przedstawić ogromną wieżę lub wieżową konstrukcję, która dominuje w krajobrazie. Ta wieża powinna mieć solidne fundamenty i wydawać się stabilna, a jednocześnie może być wyrażona w detalach, ozdobach lub futurystycznym wyglądzie.

Na szczycie tej wieży umieść mniejszą budowlę, która stanowi kolejny poziom. To może być domek, pawilon, wieżyczka lub inny rodzaj struktury. Niewielka budowla powinna harmonijnie łączyć się z większą wieżą, tworząc efekt "budowla na budowli".

Kolory i materiały powinny podkreślać kontrast między warstwami. Na przykład, większa wieża może być zbudowana z kamienia lub betonu, podczas gdy mniejsza budowla może być bardziej drewniana lub ozdobiona kolorami, które nadają jej lekkość.

Otoczenie może być pełne detali urbanistycznych lub naturalnych elementów. To może być wioska lub miasto na pierwszym planie, z elementami architektury i ludzi, tworzącego interakcję z tym nietypowym budynkiem. W tle możesz dodać krajobraz lub elementy natury, aby dodać głębi i skonstruować z monumentalnością budowli.

Uwaga: Opis ten ma charakter sugestii. Możesz dostosować go według własnej wizji artystycznej, aby stworzyć unikalny obraz ukazujący fascynującą koncepcję "budowli na budowli".



Cos nie wyszło więc znów



Uczenie Maszynowe

10 no to teraz impresjonistyczny taniec

ChatGPT

Tytuł: "Ruchy Swobodne"

Opis:

Przedstaw obraz impresjonistycznego tańca, w którym ruchy tancerzy łączą się z intensywnymi barwami i dynamicznymi szczegółami. Impresjonizm to styl artystyczny, który skupia się na chwytaniu chwilnych wrażeń i atmosfery, więc skoncentruj się na uchwyceniu energii i ruchu tańca.

Ruchy tancerzy powinny być wyraziste i swobodne. Mogą to być figury baletowe, dynamiczne ruchy modernistycznego tańca lub improwizacje, które emanują energią i emocjami.

Kolory powinny być jasne i żywe, charakterystyczne dla impresjonizmu. Wykorzystaj paletę barw, aby przedstawić zmienne odcienie światła i cienia, które tworzą dynamikę sceny. Delikatne przejścia kolorów mogą pomóc w tworzeniu efektu ruchu.

W tle możesz przedstawić scenę taneczną, taką jak sala balowa, teatr lub plener, a także publiczność obserwującą taniec. Te tła mogą być rozmyte i niedokładnie zdefiniowane, podkreślając abstrakcyjność i skupienie na ruchu tancerzy.

Użyj technik impresjonistycznych, takich jak krótkie, szybkie pociągnięcia pędzla, plamy kolorów i skomplikowane mieszanie odcieni, aby stworzyć atmosferę spontaniczności i dynamiki.

Uwaga: Opis ten ma charakter sugestii. Impresjonizm jest stylem artystycznym, który zachęca do swobodnego wyrażenia wrażeń, więc zachęcam do eksperymentowania z kolorami, formami i technikami, aby stworzyć wrażliwy obraz tanecznego ruchu.



Rozdział 2 – Kodowanie z użyciem ChatGpt

Korzystając z ChatGpt stworzyć grę w jednym z 5 języków:

1. JavaScript w przeglądarce
2. C#.
3. Java.
4. Python.
5. C++

Zadanie 5 – Proste programy

Wybrać jedną z poniższych gier:

- Kółko i krzyżyk (Tic Tac Toe): Klasyczna gra dla dwóch graczy, w której celem jest ułożenie trzech swoich symboli w linii na planszy 3x3.

```
• # Tic Tac Toe
•
• # Tworzenie planszy
• board = [' ' for _ in range(9)]
•
• def print_board():
•     print('  |  | ')
•     print(' ' + board[0] + ' | ' + board[1] + ' | ' + board[2])
•     print('  |  | ')
•     print('-----')
•     print('  |  | ')
•     print(' ' + board[3] + ' | ' + board[4] + ' | ' + board[5])
•     print('  |  | ')
•     print('-----')
•     print('  |  | ')
•     print(' ' + board[6] + ' | ' + board[7] + ' | ' + board[8])
•     print('  |  | ')
•
• def is_winner(board, player):
•     # Sprawdzanie zwycięstwa dla danego gracza
•     return (
•         (board[0] == player and board[1] == player and board[2] ==
• player) or
•         (board[3] == player and board[4] == player and board[5] ==
• player) or
•         (board[6] == player and board[7] == player and board[8] ==
• player) or
•         (board[0] == player and board[3] == player and board[6] ==
• player) or
•         (board[1] == player and board[4] == player and board[7] ==
• player) or
•         (board[2] == player and board[5] == player and board[8] ==
• player) or
•         (board[0] == player and board[4] == player and board[8] ==
• player) or
•         (board[2] == player and board[4] == player and board[6] ==
• player)
•     )
```



```

•
• def is_board_full(board):
•     # Sprawdzanie, czy plansza jest pełna
•     return ' ' not in board
•
• def play_game():
•     print("Witaj w grze Kółko i krzyżyk!")
•     print_board()
•     current_player = 'X'
•
•     while not is_board_full(board):
•         move = input("Gracz " + current_player + ", podaj numer pola
(0-8): ")
•
•         if board[int(move)] == ' ':
•             board[int(move)] = current_player
•             print_board()
•
•             if is_winner(board, current_player):
•                 print("Gracz " + current_player + " wygrał!")
•                 return
•
•             current_player = 'O' if current_player == 'X' else 'X'
•         else:
•             print("To pole jest już zajęte. Podaj inne pole.")
•
•     print("Remis!")
•
• play_game()
•

```

```

jupyter Untitled11 Last Checkpoint: 12 minut temu (autosaved)
File Edit View Insert Cell Kernel Widgets Help Trusted Python 3 (ipykernel)

Gracz O, podaj numer pola (0-8): 3
- - -
| | |
O | X |
| | |
O | | X
| | |
- - -

Gracz X, podaj numer pola (0-8): 4
- - -
| | |
O | X |
O | X | X
| | |
- - -

Gracz O, podaj numer pola (0-8): 6
- - -
| | |
O | X |
O | X | X
| | |
O | | |
- - -

Gracz O wygrał!

```

Kod wygenerowanej i działającej gry dodać do REPO na Githu umieścić w dokumencie Word plus zrzuty ekranu.

Zadanie 6 – Zaawansowane programy

Wybrać jeden z poniższych programów:

- Tetris: W tej klasycznej grze gracz musi układać spadające klocki w taki sposób, aby wypełnić poziome linie na dole ekranu.

```

• import pygame
• import random
•
• pygame.init()
•
• # Ustawienia okna gry
• screen_width = 800
• screen_height = 600
• play_width = 300
• play_height = 600
• block_size = 30
•
• top_left_x = (screen_width - play_width) // 2
• top_left_y = screen_height - play_height
•
• # Kolorowe bloki
• S = [['.....',
•      '.....',
•      '..00.',
•      '.00..',
•      '.....'],
•      ['.....',
•      '..0.',
•      '..00.',
•      '...0.',
•      '.....']]
•
• Z = [['.....',
•      '.....',
•      '..00.',
•      '..00.',
•      '.....'],
•      ['.....',
•      '..0.',
•      '..00.',
•      '.0...',
•      '.....']]
•
• I = [['.....',
•      '..0.',
•      '..0.',
•      '..0.',
•      '..0.'],

```

```

•      [ '.....',
•      '0000.',
•      '.....',
•      '.....',
•      '.....' ] ]
•
•  0 = [ [ '.....',
•      '.....',
•      '.00.',
•      '.00.',
•      '.....' ] ]
•
•  J = [ [ '.....',
•      '.0...',
•      '.000.',
•      '.....',
•      '.....' ],
•      [ '.....',
•      '..00.',
•      '..0...',
•      '..0...',
•      '.....' ],
•      [ '.....',
•      '.000.',
•      '...0.',
•      '.....' ],
•      [ '.....',
•      '..0...',
•      '..0...',
•      '.00.',
•      '.....' ] ]
•
•  L = [ [ '.....',
•      '...0.',
•      '.000.',
•      '.....',
•      '.....' ],
•      [ '.....',
•      '..0...',
•      '..0...',
•      '..00.',
•      '.....' ],
•      [ '.....',
•      '.000.',
•      '.0...',
•      '.....' ],

```

```

•      ['.....',
•      '..00..',
•      '..0..',
•      '..0..',
•      '.....']]
•
•  T = [['.....',
•      '..0..',
•      '.000.',
•      '.....',
•      '.....'],
•      ['.....',
•      '..0..',
•      '..00.',
•      '..0..',
•      '.....'],
•      ['.....',
•      '.....',
•      '.000.',
•      '..0..',
•      '.....'],
•      ['.....',
•      '..0..',
•      '.00..',
•      '..0..',
•      '.....']]
•
•  shapes = [S, Z, I, O, J, L, T]
•  shape_colors = [(0, 255, 0), (255, 0, 0), (0, 255, 255), (255, 255, 0),
•  (255, 165, 0), (0, 0, 255), (128, 0, 128)]
•
•  class Piece(object):
•      rows = 20
•      columns = 10
•
•      def __init__(self, column, row, shape):
•          self.x = column
•          self.y = row
•          self.shape = shape
•          self.color = shape_colors[shapes.index(shape)]
•          self.rotation = 0
•
•      def create_grid(locked_positions={}):
•          grid = [[(0, 0, 0) for _ in range(Piece.columns)] for _ in
•          range(Piece.rows)]
•
•          for row in range(len(grid)):
•              for col in range(len(grid[row])):

```

```

•         if (col, row) in locked_positions:
•             color = locked_positions[(col, row)]
•             grid[row][col] = color
•
•     return grid
•
• def convert_shape_format(piece):
•     positions = []
•     shape_format = piece.shape[piece.rotation % len(piece.shape)]
•
•     for i, row in enumerate(shape_format):
•         for j, col in enumerate(list(row)):
•             if col == '0':
•                 positions.append((piece.x + j, piece.y + i))
•
•     for i, pos in enumerate(positions):
•         positions[i] = (pos[0] - 2, pos[1] - 4)
•
•     return positions
•
• def valid_space(piece, grid):
•     accepted_positions = [[(j, i) for j in range(Piece.columns) if
grid[i][j] == (0, 0, 0)] for i in range(Piece.rows)]
•     accepted_positions = [j for sub in accepted_positions for j in sub]
•
•     formatted = convert_shape_format(piece)
•
•     for pos in formatted:
•         if pos not in accepted_positions:
•             if pos[1] > -1:
•                 return False
•
•     return True
•
• def check_lost(positions):
•     for pos in positions:
•         x, y = pos
•         if y < 1:
•             return True
•
•     return False
•
• def get_shape():
•     return Piece(5, 0, random.choice(shapes))
•
• def draw_text_middle(text, size, color, surface):
•     font = pygame.font.SysFont(None, size)
•     label = font.render(text, True, color)

```



```

• def update_score(nscore):
•     with open('scores.txt', 'r') as f:
•         lines = f.readlines()
•         score = int(lines[0].strip())
•
•         if nscore > score:
•             with open('scores.txt', 'w') as f:
•                 f.write(str(nscore))
•
• def draw_window(surface, grid, score=0, last_score=0):
•     surface.fill((0, 0, 0))
•
•     pygame.font.init()
•     font = pygame.font.SysFont(None, 60)
•     label = font.render('TETRIS', True, (255, 255, 255))
•
•     surface.blit(label, (top_left_x + play_width / 2 -
• (label.get_width() / 2), 30))
•
•     # Aktualny wynik
•     font = pygame.font.SysFont(None, 30)
•     label = font.render('Wynik: ' + str(score), True, (255, 255, 255))
•
•     sx = top_left_x + play_width + 50
•     sy = top_left_y + play_height / 2 - 100
•
•     surface.blit(label, (sx + 20, sy + 160))
•
•     # Poprzedni wynik
•     label = font.render('Rekord: ' + str(last_score), True, (255, 255,
255))
•
•     sx = top_left_x - 200
•     sy = top_left_y + 200
•
•     surface.blit(label, (sx + 20, sy + 160))
•
•     for i in range(len(grid)):
•         for j in range(len(grid[i])):
•             pygame.draw.rect(surface, grid[i][j], (top_left_x + j *
block_size, top_left_y + i * block_size, block_size, block_size), 0)
•
•     pygame.draw.rect(surface, (255, 0, 0), (top_left_x, top_left_y,
play_width, play_height), 5)
•
•     pygame.display.update()
•
• def main():

```

```

•     locked_positions = {}
•     grid = create_grid(locked_positions)
•
•     change_piece = False
•     run = True
•     current_piece = get_shape()
•     next_piece = get_shape()
•     clock = pygame.time.Clock()
•     fall_time = 0
•     fall_speed = 0.27
•     level_time = 0
•     score = 0
•
•     while run:
•         grid = create_grid(locked_positions)
•         fall_time += clock.get_rawtime()
•         level_time += clock.get_rawtime()
•         clock.tick()
•
•         if level_time / 1000 > 5:
•             level_time = 0
•             if fall_speed > 0.12:
•                 fall_speed -= 0.005
•
•         if fall_time / 1000 > fall_speed:
•             fall_time = 0
•             current_piece.y += 1
•             if not (valid_space(current_piece, grid)) and
current_piece.y > 0:
•                 current_piece.y -= 1
•                 change_piece = True
•
•         for event in pygame.event.get():
•             if event.type == pygame.QUIT:
•                 run = False
•                 pygame.display.quit()
•                 quit()
•
•             if event.type == pygame.KEYDOWN:
•                 if event.key == pygame.K_LEFT:
•                     current_piece.x -= 1
•                     if not valid_space(current_piece, grid):
•                         current_piece.x += 1
•
•                 elif event.key == pygame.K_RIGHT:
•                     current_piece.x += 1
•                     if not valid_space(current_piece, grid):
•                         current_piece.x -= 1

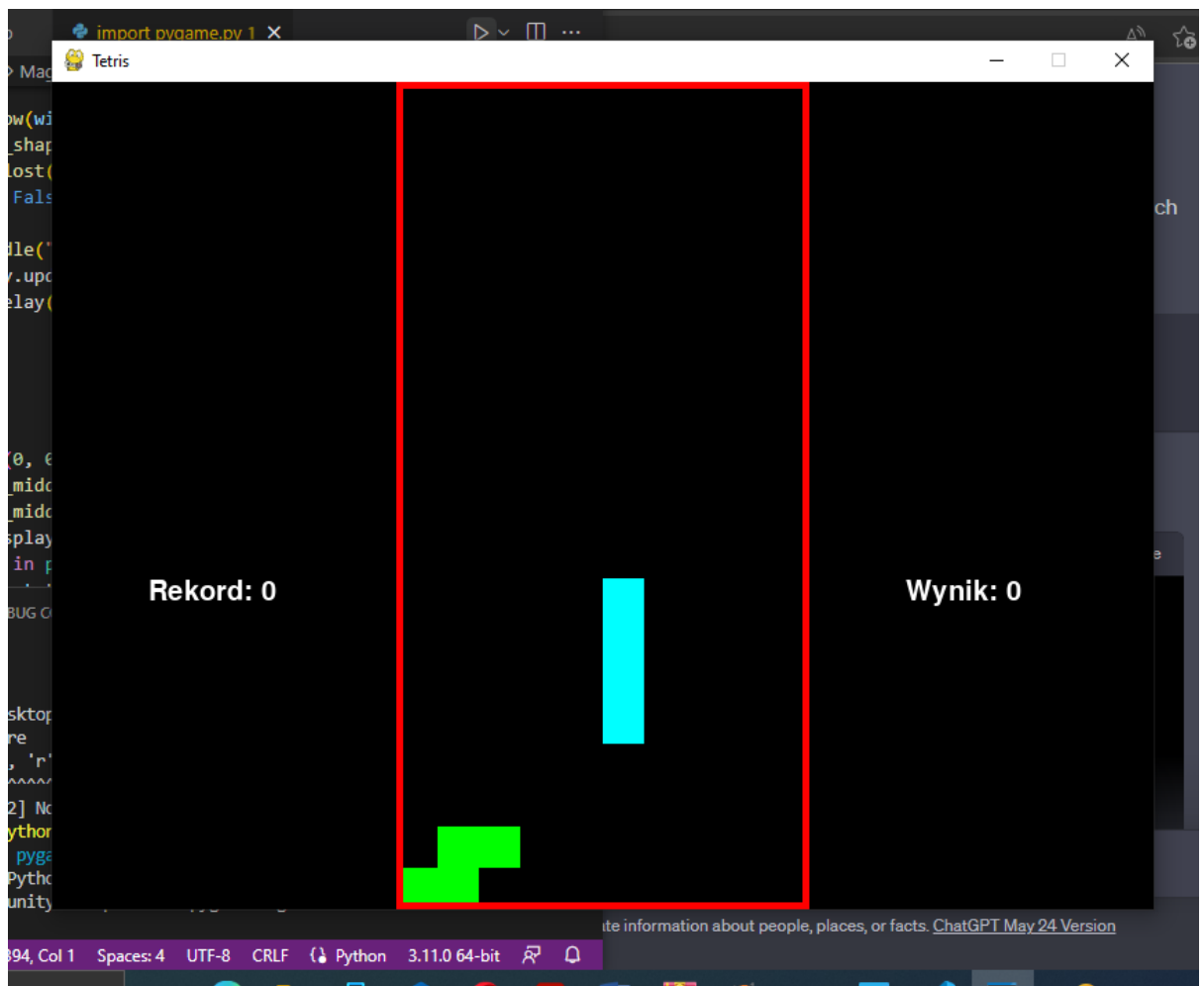
```

```

•
•         elif event.key == pygame.K_DOWN:
•             current_piece.y += 1
•             if not valid_space(current_piece, grid):
•                 current_piece.y -= 1
•
•         elif event.key == pygame.K_UP:
•             current_piece.rotation += 1
•             if not valid_space(current_piece, grid):
•                 current_piece.rotation -= 1
•
•
•     shape_pos = convert_shape_format(current_piece)
•
•     for i in range(len(shape_pos)):
•         x, y = shape_pos[i]
•         if y > -1:
•             grid[y][x] = current_piece.color
•
•     if change_piece:
•         for pos in shape_pos:
•             p = (pos[0], pos[1])
•             locked_positions[p] = current_piece.color
•
•     current_piece = next_piece
•     next_piece = get_shape()
•     change_piece = False
•     score += clear_rows(grid, locked_positions)
•
•     if score > 0:
•         update_score(score)
•
•     draw_window(win, grid, score)
•     draw_next_shape(next_piece, win)
•     if check_lost(locked_positions):
•         run = False
•
•     draw_text_middle("Przegrana!", 80, (255, 255, 255), win)
•     pygame.display.update()
•     pygame.time.delay(2000)
•
• def main_menu():
•     run = True
•     while run:
•         win.fill((0, 0, 0))
•         draw_text_middle('TETRIS', 60, (255, 255, 255), win)
•         draw_text_middle('Naciśnij spację, aby zagrać', 40, (255, 255,
255), win)
•         pygame.display.update()

```

- `for event in pygame.event.get():`
- `if event.type == pygame.QUIT:`
- `run = False`
- `if event.type == pygame.KEYDOWN:`
- `if event.key == pygame.K_SPACE:`
- `main()`
-
- `pygame.quit()`
-
- `win = pygame.display.set_mode((screen_width, screen_height))`
- `pygame.display.set_caption('Tetris')`
-
- `main_menu()`
-



Kod wygenerowanej i działającej gry dodać do REPO na Githu umieścić w dokumencie Word plus zrzuty ekranu.

Rozdział 3 - Ocenianie

Sprawozdanie

1. Wykonać jedno z siedmiu zadań na ocenę co najmniej dst.
2. Wynik w postaci linku do repozytorium na GitHub oraz sprawozdanie w PDF. Sprawozdanie w PDF powinno się nazywać:
ProgramowanieObiektowe_Zadanie4_Rozwiazanie_ImieNazwisko_XXXXX.pdf, gdzie ImieNazwisko zastąpić swoim imieniem i nazwiskiem a XXXXX numerem albumu.
3. Rozwiązanie powinno być linkiem do repo na GitHub, gdzie będzie pokazane PDFy albo BMP w wysokiej rozdzielczości z diagramami i wskazaniem w nazwie, którego zadania dotyczą.

Wyniki wysłać na adres leslaw.pawlaczyk@chorzow.wsb.pl jako dokument w Word.

