Sprawozdanie z Projektu

OBRAZKOWE TETRIS

nr projektu – **20**

Podstawy Grafiki Komputerowej

Semestr 4
Informatyka Stosowana
Wydział Fizyki i Informatyki Stosowanej
Akademia Górniczo-Hutnicza
im. Stanisława Staszica
w Krakowie

Daria Kokot
Alicja Kałuża
Kacper Duda

26 czerwca 2023

1. Tytuł projektu i jego autorzy

Obrazkowe Tetris – projekt o numerze 20. Wymagania projektu [LINK] [ostatni dostęp 21.06.2023 r.]

Trzyosobowy zespół specjalistów IT wraz z podziałem obowiązków:

- Daria Kokot – odpowiedzialna za utworzenie layoutu i grafiki aplikacji, implementację funkcji do obsługi zdarzeń oraz testowanie kolejnych funkcjonalności aplikacji w celu zapewnienia ich poprawnego działania,
- Alicja Kałuża – odpowiedzialna za zaprojektowanie struktury kodu programu oraz matematyczne opracowanie algorytmów obliczeniowych użytych w projekcie, a także za ich skuteczną implementację,
- Kacper Duda – prace koncepcyjne (logika pętli), projektowanie layoutu, pisanie dokumentacji, dokumentacja techniczna (Doxygen), poprawa komentarzy w kodzie.

2. Opis projektu

Obrazkowe Tetris (dalej gra) to prosta gra wzorowana na kultowej grze Tetris. Gra pozwala na ułożenie obrazka kolorowego na podstawie wzorca na wcześniej przygotowanym wyszarzanym tle. Obrazek dzielony jest na kwadraty, których ilość w danym wierszu jest ustalana przez gracza za pośrednictwem menu. Istnieje wybór obrazka na tło z podanej listy oraz możliwość dynamicznego ustalania prędkości spadania kwadratów. Każdy kwadrat może być obracany i odbijany. Gra kończy się po ułożenia całego obrazka, a wynik jest ustalany na podstawie czasu gry.

3. Założenia wstępne przyjęte w realizacji projektu oraz nasza realizacja

Wymagania podstawowe:

- Program powinien pozwalać na wybór jednego z kilku zdefiniowanych obrazków – spełnione,
- Powinna również istnieć możliwość ustalenia na ile kwadratów obrazek ma zostać podzielony oraz tempa ich ruchu – spełnione,
- Kwadraty mogą przesuwać się w dół skokowo o odległość równą bokowi kwadratu – na rzecz wymagań rozszerzonych ruch został upłynniony.

Wymagania rozszerzone:

- gwarancja w miarę płynnego ruchu kwadratów – spełnione,
- kwadraty mogą być również odbite wzdłuż pionowej lub poziomej osi – spełnione,
- obraz wzorcowy mógłby być wyświetlany w studni w postaci czarno białej, a spadające kwadraty budowałyby go w wersji kolorowej - spełnione,

Uwaga dodatkowa – zadbano, by spadający kwadrat zawsze dał się ulokować we właściwym miejscu, nie jest wymagana idealna precyzja (tzw. pixel perfect), kwadrat musi się znaleźć w pewnym obszarze w kształcie kwadratu o większym boku niż ten kwadrat, który spada.

Podsumowując, uważamy, że wszystkie wymagania projektu zostały spełnione.

4. Analiza projektu

4.1. Specyfikacja danych wejściowych

Gra posiada następujące wejście od gracza:

a) opcje typu radio button do wyboru:
   - istnieje wybór obrazka w tle z podanej listy
   - ilości kwadratów na wiersz
   - wybór tempa rozgrywki (prędkość spadania klocka)

b) wejście z klawiatury
   - klawisze „strzałki" do poruszania w prawo i lewo
   - F1 – do odbijania w pionie
   - tab – do odbijania w poziomie
   - strzałka w górę do odwracania w lewo, a w dół – w prawo

Gra nie wczytuje żadnych plików.

4.2. Opis oczekiwanych danych wyjściowych

Dane wyjściowe gry Tetris zwykle obejmują następujące elementy:

- plansza – obszar gry, na którym są wyświetlane spadające klocki, plansza zazwyczaj ma różne wymiary w zależności od opcji wybranych przez gracza,
- spadające klocki – główne elementy gry Tetris, są to kwadraty zawierające wycinek bazowego obrazka,
- wynik – jest wyświetlany na ekranie i zwiększa się w miarę upływu czasu, im niższy czas, tym lepsze osiągnięcie,
- poziom trudności – gracz może zmienić poziom trudności ustalając liczbę kafelków w rzędzie oraz szybkość spadania klocków.
- czas gry – może być wyświetlany na ekranie i mierzy czas, jaki upłynął od rozpoczęcia gry.

4.3. Zdefiniowanie struktur danych

Program podzielony jest na pomniejsze struktury danych:

- Struktura **Square** odpowiedzialna za przechowywanie informacji o danym kwadracie, który został już odpowiednio ułożony – jego położenie i wewnętrzna grafika,
- klasa **GUIMyFrame1** pozwala uporządkować layout – ułożenie wszystkich elementów na ekranie oraz obsługę zdarzeń. Między innymi:
    - obsługę wciśnięcia klawiszy,
    - obsługę wybrania z listy odpowiednich opcji,
    - obsługę zmiany rozmiaru okna,
    - obsługę zdarzeń zegara gry,
    - przechowywanie obiektów klasy **Square** w wektorze,
    - sprawdza w metodzie *Game*, czy warunki zakończenia gry są spełnione i zajmuje się wyświetleniem komunikatu,
- klasa **MyFrame1** stanowi łącznik między wxWidgets i logiką aplikacji, a mianowicie:
    - za pomocą biblioteki uruchamia okno aplikacji i ustala jego początkowe właściwości – tytuł, przyciski, początkowy rozmiar,
    - gospodaruje przestrzenią okna – dzieli je na trzy kolumny i dynamicznie dostosowuje ich wysokość i szerokość w zależności od rozmiaru okna ustalonego przez użytkownika,

- tworzy dwa płótna (*wxPanel*) i oddaje je do dyspozycji innej części programu odpowiedzialnej za rysownie kwadratów na nich,
- umieszcza elementy interfejsu (przycisk start, wybór tła, wybór ilości kwadratów w wierszu, tempo gry),
- przekazuje decyzje gracza do dalszego rozpracowania (wywołuje metody obsługujące wydarzenia w klasie **GUIMyFrame1**).

4.4. Specyfikacja interfejsu użytkownika

Okno jest podzielone na 3 kolumny.

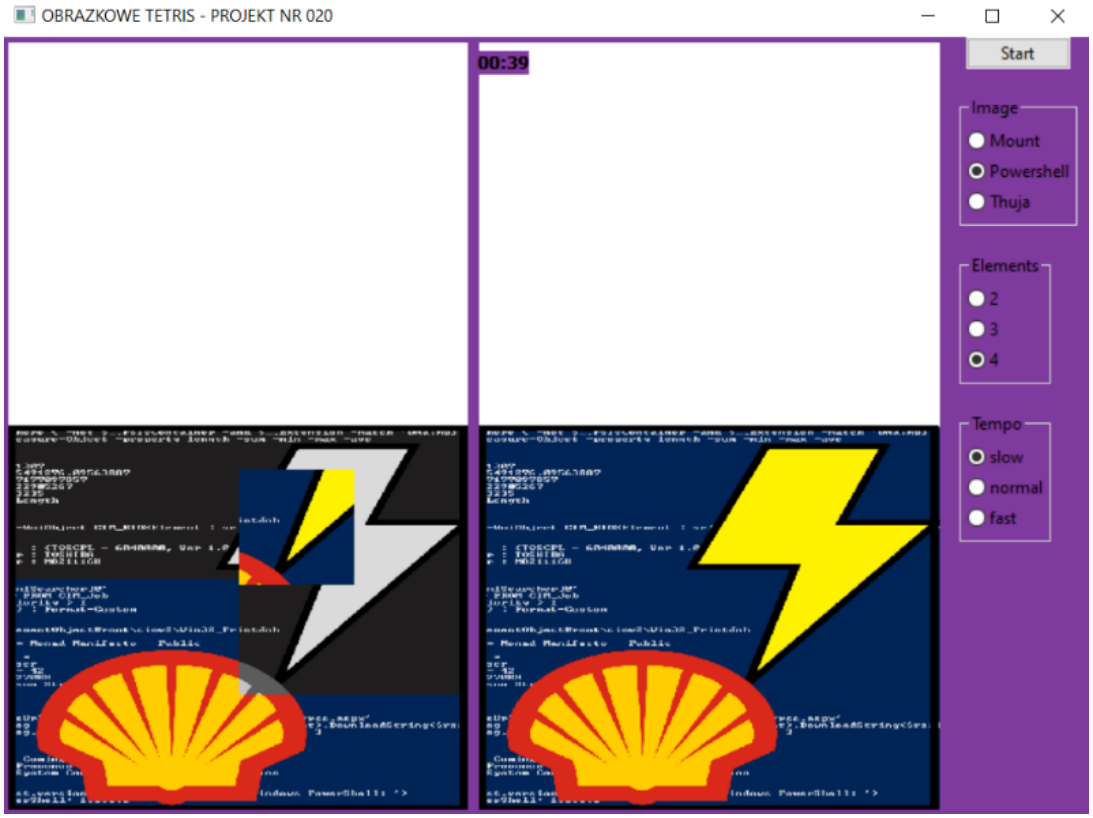
Pierwsze dwie z lewej mają tę samą szerokość i stanowią płótno dla obrazków:

- pierwsza z lewej to płótno, które na początku jest wyszarzaną wersją tła, następnie jest powoli nadpisywane (zastępowane kwadratami kolorowymi)
- środkowa stanowi statyczne (w sensie rozgrywki, bo dostosowuje się do rozmiaru okna) kompletny kolorowy obrazek

Ostatnia kolumna zawiera:

- przycisk do rozpoczęcia rozgrywki
- wybór tła
- wybór tempa rozgrywki
- ilość kwadratów na wiersz

Poniżej znajdują się przykłady, jak wygląda rozkład elementów na stronie:



Zrzut ekranu 1 – tło *PowerShell*

Zrzut ekranu 2 – tło *Góry*

4.5. Wyodrębnienie i zdefiniowanie zadań

Program (logicznie) jest podzielony na dwie zasadnicze części:

- część odpowiedzialna za interfejs,
    - inicjalizacja układu graficznego i utrzymanie go pomimo zmiany rozmiaru okna
    - przekazywanie wejścia gracza do części odpowiedzialna za rozgrywkę,
    - udostępnianie wygodnego sposobu do części odpowiedzialnej za rozgrywkę, by na płótnach można było rysować,
- część odpowiedzialna za rozgrywkę,
    - obsługa zdarzeń zainicjowanych przez użytkownika,
    - przechowuje wewnętrzny stan gry
        - zegar
        - ułożone kwadraty
        - aktualną pozycję spadającego kwadratu

4.6. Decyzja o wyborze narzędzi programistycznych

Środowiskiem programistycznym jest Microsoft Visual Studio 2022, które zapewnia wygodę użytkowania:

- podświetlanie i kolorowanie kodu,
- podpowiadanie nazw klas i obiektów w trakcie pisania,
- pomoc w imporcie bibliotek – podpowiada nazwy plików nagłówkowych.

Jedyną biblioteką (frameworkiem) wykorzystywaną przez nas jest wxWidgets, która zapewnia ogromną pomoc w wyświetlaniu elementów, jej funkcje to:

- szeroki zakres gotowych komponentów interfejsu użytkownika, takich jak przyciski, pola tekstowe, listy rozwijane, drzewa, okna dialogowe itp.,
- funkcje do obsługi zdarzeń, rysowania grafiki, obsługi plików, komunikacji sieciowej i wiele innych,
- tworzenie aplikacji wieloplatformowych przy użyciu jednego kodu,
- natywny wygląd i zachowanie na różnych systemach operacyjnych,
- ułatwienie rozwijania aplikacji graficznych,
- dostępność dla wielu języków programowania,
- aktywna społeczność programistów i bogata dokumentacja,
- popularność wśród twórców oprogramowania wieloplatformowego.

5. Podział pracy i analiza czasowa

Uważamy, że każdy członek zespołu wykonał równą ilość pracy i włożył równą ilość czasu w ten projekt.

Podział pracy wyglądał następująco:

Daria Kokot:

- utworzenie layoutu i grafiki,
- implementacja funkcji do obsługi zdarzeń,
- testowanie kolejnych funkcjonalności aplikacji.

Alicja Kałuża:

- zaprojektowanie struktury kodu programu,
- matematyczne opracowanie algorytmów obliczeniowych użytych w programie i ich implementacja.

Kacper Duda:

- prace koncepcyjne,
- prace projektowe,
- pisanie dokumentacji.

Analiza głównych bloków zadań

1. Pierwsze etapy (łącznie – 5h)
    a. Zapoznanie się z możliwościami platformy GitHub i utworzenie repozytorium (0.5h)
    b. Wybór i dostosowanie odpowiedniego środowiska programistycznego do projektu (1h)
    c. Opracowanie krótkiego opisu projektu (0.5h)
    d. Określenie głównych koncepcji projektowych (3h)
2. Prace koncepcyjne (łącznie – 17h)
    a. Projektowanie interfejsu użytkownika (1h)
    b. Analiza podstawowych i zaawansowanych wymagań projektu (1h)
    c. Badanie koncepcyjne autorskich funkcji programu (3h)
    d. Tworzenie klas występujących w kodzie źródłowym projektu (2h)
    e. Opracowanie matematycznych algorytmów (10h)
3. Tworzenie interfejsu użytkownika (łącznie - 9h)
    a. Tworzenie wstępnego, ograniczonego interfejsu użytkownika (3h)
    b. Stopniowe dodawanie nowych funkcji (4h)
    c. Ręczne testowanie wszystkich elementów interfejsu użytkownika (2h)
4. Obsługa wczytywanych plików (łącznie – 2h)
    a. Tworzenie grafiki i implementacja jej obsługi (2h)
5. Opracowanie i implementacja algorytmów ruchu obiektów oraz logiki gry (łącznie – 14h)
    a. Analiza poruszania się kafelków (5h)
    b. Implementacja funkcji poruszania i przekształcania kafelków (4h)
    c. Implementacja funkcji obsługi zdarzeń (5h)
6. Debugowanie programu (łącznie - 13h)
    a. Wykrywanie błędów (3h)
    b. Wprowadzanie drobnych poprawek algorytmicznych (4h)
    c. Poprawki w działaniu interfejsu użytkownika (2h)
    d. Refaktoryzacja kodu (4h)
7. Testowanie programu (łącznie - 6h)
    a. Testowanie programu w warunkach użytkowych (6h)
8. Opracowywanie dokumentacji (łącznie - 11h)
    a. Tworzenie dokumentacji w trakcie pracy nad projektem (4h)
    b. Tworzenie i utrzymanie strony internetowej z dokumentacją projektu (1h)
    c. Końcowe poprawki dokumentacji działającego programu (3h)
    d. Generowanie dokumentacji przy użyciu narzędzia Doxygen (3h)
9. Sprawozdanie z projektu (łącznie – 23h)
    a. Opracowanie wstępnego szablonu sprawozdania (6h)
    b. Edycja poszczególnych części sprawozdania (12h)

c. Dodawanie grafik, schematów i ich opisów (2h)
d. Korekty merytoryczne i stylistyczne (3h)

Szacowany łączny nakład pracy wynosi około 90h godzin. Praca była podzielona równo między członków zespołu.

6. Opracowanie i opis niezbędnych algorytmów

Najważniejszym elementem gry jest jej pętla oraz jej poszczególne iteracje zwane tickami. Pętla wykonuje się do momentu spełnienia końcowych.

Po rozpoczęciu gry procedura DoImageTab() wypełnia tablicę kafelków elementami o odpowiednim rozmiarze oraz w odpowiedniej ilości.

Następnie uruchamiany jest zegar. W każdym tiku wywoływana jest procedura Game(). Wewnątrz tej procedury odbywa się spadanie kafelków. Na samym początku sprawdzamy czy wektor zawierający prawidłowo ułożone kwadraty zawiera wszystkie tworzące obrazek. Jeśli tak, gra kończy się i na ekranie pokazuje się okienko z informacją o wygranej oraz o liczbie zdobytych punktów (czas).

Jeśli nie zatrzymujemy zegara i sprawdzamy czy i jaki klawisze zostały wciśnięte – wtedy zmieniamy poszczególne zmienne.

Jeśli klawisze strzałek w lewo lub w prawo- zmienna odpowiadająca przesunięciu, jeśli klawisze strzałek w górę lub w dół, to zmienna rotacji, jeśli klawisz TAB lub F1 – zmienna odbicia.

Sprawdzamy, czy zmienna odpowiadająca za przesunięcie kafelka w poziomie jest równa 0. Jeśli tak to pobieramy nowy element z tablicy kafelków z indeksami row i col. Losujemy liczbę odpowiadającą którą ustalamy na liczbę rotacji.

Kopiujemy go to obrazka tymczasowego. Następnie rysujemy go na lewym panelu o współrzędnych: x – połowa ekranu + zmienna przesunięcia, y – zmienna przesunięcia w poziomie.

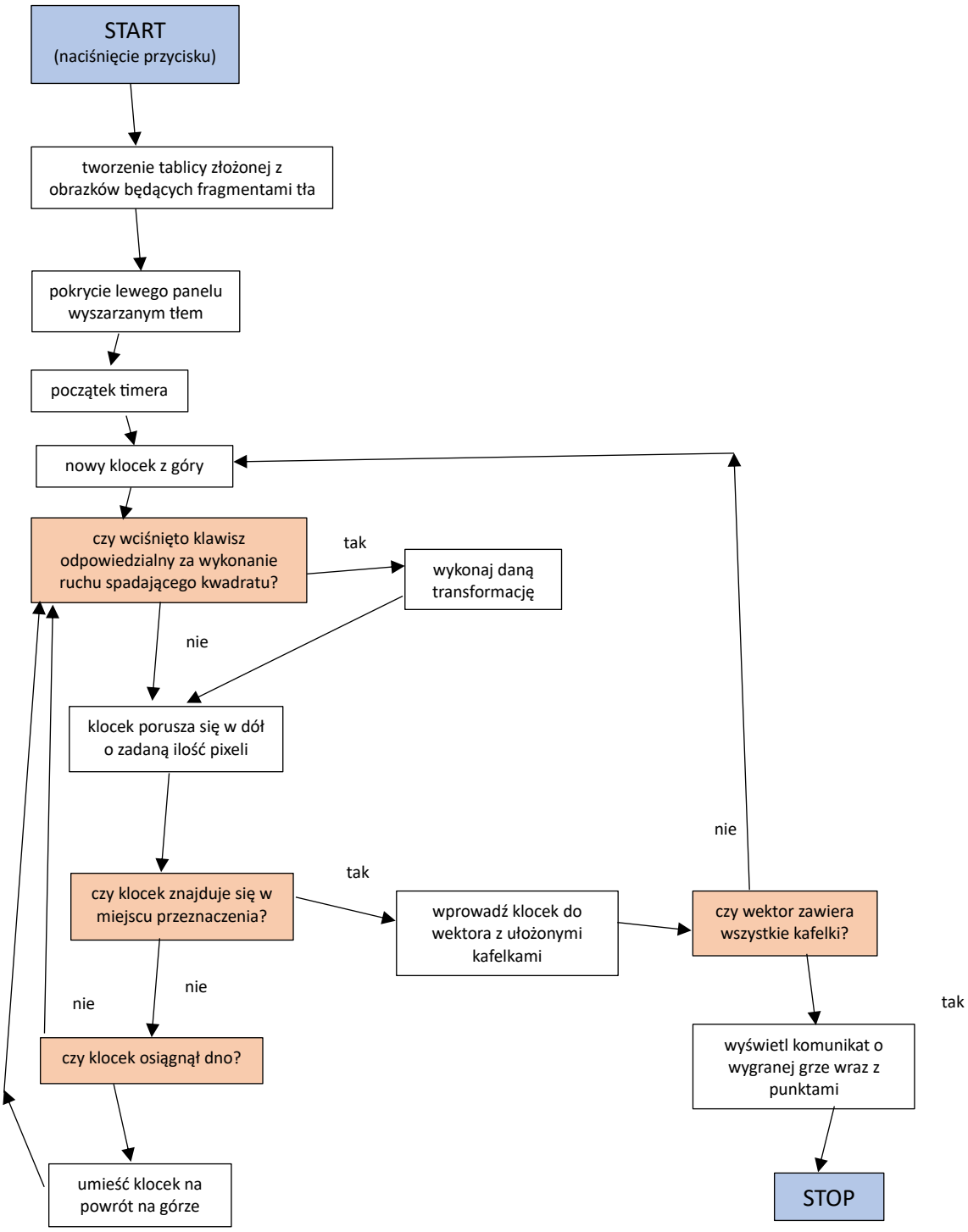
Sprawdzamy, która ze zmiennych odpowiadających za wciśnięcie któregoś z klawiszy jest inna od zera. W zależności, która z nich wykonujemy jedną z operacji: zmiana współrzędnych wyświetlenia kafelka w lewo lub w prawo, odbicie lustrzane bądź rotacja o odpowiedni kąt.

Kolejną rzeczą jest sprawdzenie czy kafelek znajduje się na pierwotnym miejscu - jeśli tak, to ustawiamy wszystkie zmienne na zero oraz wkładamy kafelek do wektora, inkrementujemy wartość indeksu (col) w jakim pobierzemy element.

Sprawdzamy czy zmienna col przekroczyła liczby kolumn w tablicy. Jeśli tak, inkrementujemy wartość indeksu (row), a col ustawiamy na zero.

Jeśli warunek nie zachodzi inkrementujemy wartość zmiennej przesunięcia w osi y. Następnie sprawdzamy, czy ta zmienna nie przekracza wysokości panelu. Jeśli tak, ustawiamy ją na zero (ten sam kwadrat spada z góry jeszcze raz). Przechodzimy do kolejnego tiku.

Schemat blokowy logiki



7. Kodowanie

Dokumentacja wygenerowana w Doxygen (załączona do tej dokumentacji) zawiera opis klas, metod oraz hierarchię klas.

Aby otworzyć dokumentację wygenerowaną przez Doxygen, należy otworzyć w przeglądarce plik o nazwie *DOC/Doxygen/html/index.html*

8. Testowanie

Nie przeprowadzaliśmy testów na poziomie kodu programu – nie pisaliśmy testów jednostkowych ani innych. Uznaliśmy, że wielkość projektu jest za mała, aby pisanie testów było kwestią krytyczną.

Jednakże każdy fragment programu został empirycznie sprawdzony poprzez wielokrotną rozgrywkę.

9. Wdrożenie, raport i wnioski

Grę udało się przedyskutować w zespole, zaprojektować, napisać i skompilować do postaci wykonywalnej (.exe) na komputerach z systemem Windows z architekturą procesora 64-bitową.

Wszystkie pliki przekazywane w ramach projektu są skompresowane oraz podzielone na foldery:

- *BIN* – zawiera plik wykonywalny, pliki bibliotek .dll oraz grafiki tła
- *SRC* – pliki źródłowe (.h, .cpp, itd.)
- *DOC* – pliki związane z dokumentacją (ten dokument, dokumentacja wygenerowana w Doxygen)

Do zrobienia grafiki wykorzystano zdjęcia ze stron:
https://www.looper.com/img/gallery/whatever-happened-to-the-sun-baby-from-teletubbies/l-intro-1684946032.jpg
https://en.wikipedia.org/wiki/PowerShell