Specyfikacja Projektu

Opis ogólny

Celem projektu jest stworzenie aplikacji desktopowej w języku Python, która pomaga użytkownikowi w układaniu puzzli cyfrowych. Program umożliwia użytkownikowi wgranie obrazu, a następnie oferuje mu różne możliwości podziału go na kawałki.

W wersji minimalnej, program układa podzielony obraz i pokazuje użytkownikowi efekt końcowy.

W wersji rozbudowanej, tworzymy asystenta układania puzzli, który będzie podpowiadał użytkownikowi następne posunięcia. (Algorytm policzy na początku cały obraz i na podstawie tego będzie podpowiadał).

Działanie programu

Wersja minimalna

- Użytkownik wczytuje obraz wejściowy.
- Użytkownik wybiera jak dzielony jest obraz. (Ilość elementów, wybór z dostępnych siatek. UWAGA: Puzzle są dobrze zorientowane)
- Program oblicza (wielowątkowo) ułożenie puzzli.
- Program sprawdza (wielowątkowo) poprawność ułożenia. (Porównuje, obraz początkowy z końcowym)
- Program pokazuje użytkownikowi wynik obliczeń.

Wersja rozbudowana

- Użytkownik wczytuje obraz wejściowy.
- Użytkownik wybiera jak dzielony jest obraz. (Ilość elementów, wybór z dostępnych siatek UWAGA: Puzzle są dobrze zorientowane)
- Użytkownik rozpoczyna grę
- Program oblicza (wielowątkowo) ułożenie puzzli.
- Program sprawdza (wielowątkowo) poprawność ułożenia. (Porównuje, obraz początkowy z końcowym)
- Użytkownik dopasowuje puzzle. (program sprawdza poprawność ruchów)
- Użytkownik może prosić o podpowiedzi. (Prosi o pomoc w dopasowaniu puzzla na dane miejsce, Prosi o podpowiedzenie kilku potencjalnych puzzli?)
- Gra toczy się do ułożenia obrazka.

Pomysły na dalszy rozwój

- Możliwość obracania puzzli
- Dodatkowe poziomy podpowiedzi
- Porównywanie po obrazie

Algorytm

Poziom 1: Równoległe sprawdzanie kandydatów dla jednej pozycji

Cel: Przyspieszenie dopasowania puzzla do pojedynczej pozycji w siatce.

Jak to działa:

1. Dla wybranej pozycji w siatce (np. (2,1)), do której chcemy dobrać puzzel:
    a. Program zna już sąsiednie puzzle (np. lewy lub górny sąsiad) i ich krawędzie.
2. Program tworzy zbiór dostępnych puzzli (jeszcze nieużytych).
3. Każdy dostępny puzzel jest przypisany do jednego z wielu wątków, gdzie jest sprawdzany równolegle pod kątem dopasowania do tej pozycji:
    a. Sprawdza się zgodność krawędzi (np. prawa krawędź puzzla po lewej == lewa krawędź kandydata).
4. Wszystkie wyniki są zbierane, a najlepiej pasujący puzzel wybierany do umieszczenia na tej pozycji.

Efekt: Czas przeszukiwania i dopasowania puzzla skraca się, ponieważ wszystkie kandydaty są oceniane równolegle, zamiast po kolei.

Poziom 2: Równoległa analiza wielu pozycji w siatce

Cel: Układanie większej ilości puzzli w tym samym czasie.

Jak to działa:

1. Program utrzymuje kolejkę pozycji do obsłużenia – to są pola w siatce, które mają co najmniej jednego sąsiada i są jeszcze puste.
2. W jednej „turze” (czyli jednej iteracji głównej pętli), z kolejki pobieranych jest kilka takich pozycji.
3. Każda z tych pozycji przetwarzana jest równolegle:
    a. Dla każdej pozycji uruchamiane są niezależne wątki, które wykonują całą procedurę z poziomu 1 (czyli sprawdzanie kandydatów i wybór najlepszego puzzla).
4. Po zakończeniu tej tury, wszystkie dopasowane puzzle są wpisywane do siatki.
5. Na postawie nowo ułożonych puzzli, dodawane są nowe sąsiednie pozycje do kolejki.

Efekt: Układanka rozbudowywana jest w wielu miejscach na raz – nie tylko jeno pole na raz, ale np. kilka równoległych „odgałęzień”.

Synchronizacja i spójność danych:

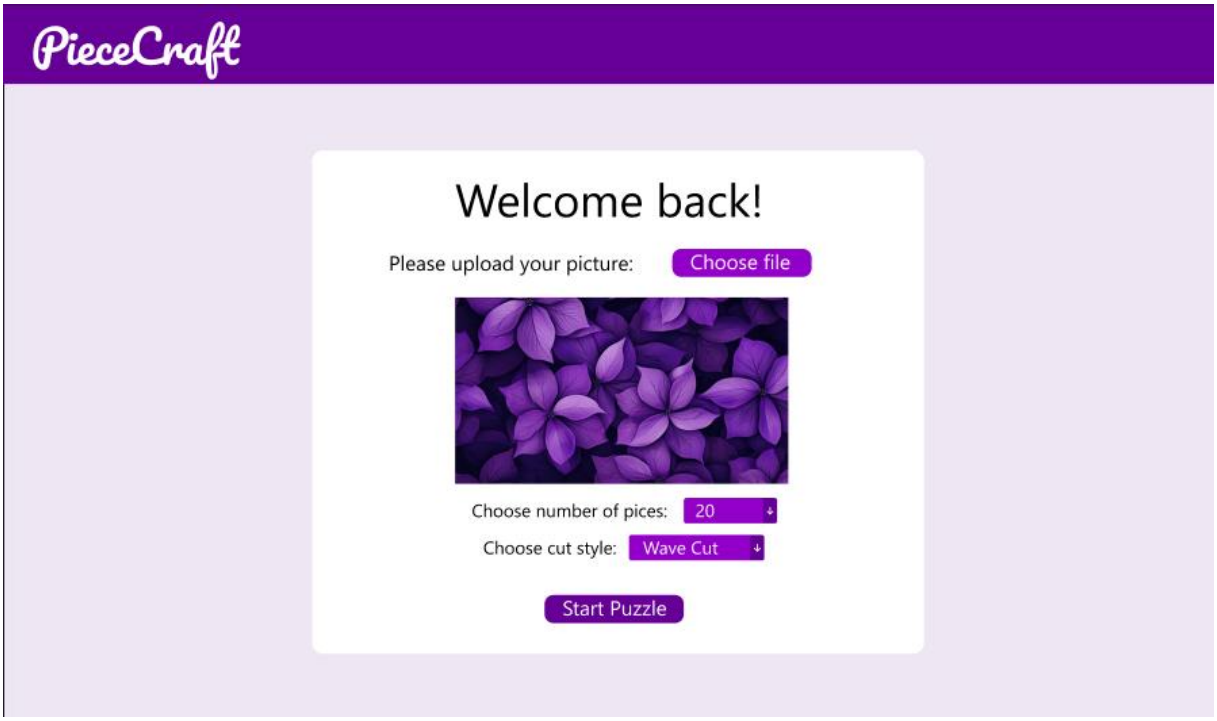
Ponieważ wiele pozycji i kandydatów jest sprawdzanych jednocześnie, potrzebne jest:

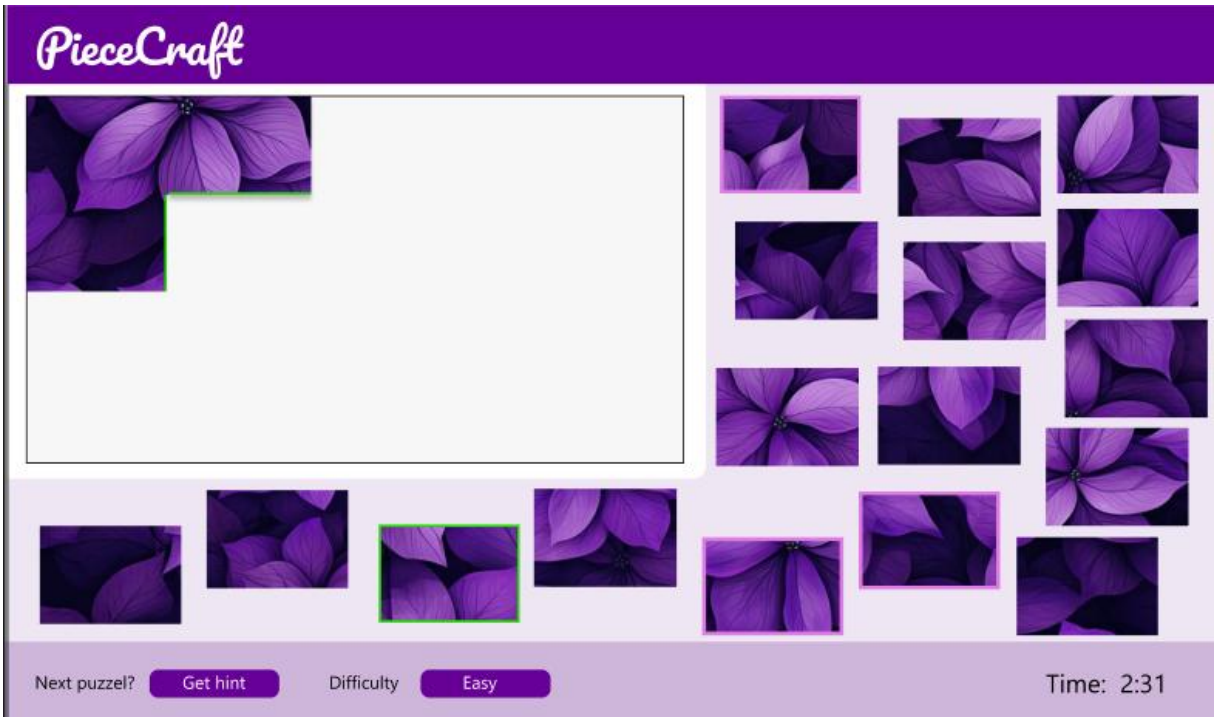
1. Skoordynowana obsługa listy dostępnych puzzli – aby nie dopasować tego samego puzzla w dwóch miejscach,
2. Synchronizacja przy aktualizacji siatki i usuwaniu puzzli z puli dostępnych.

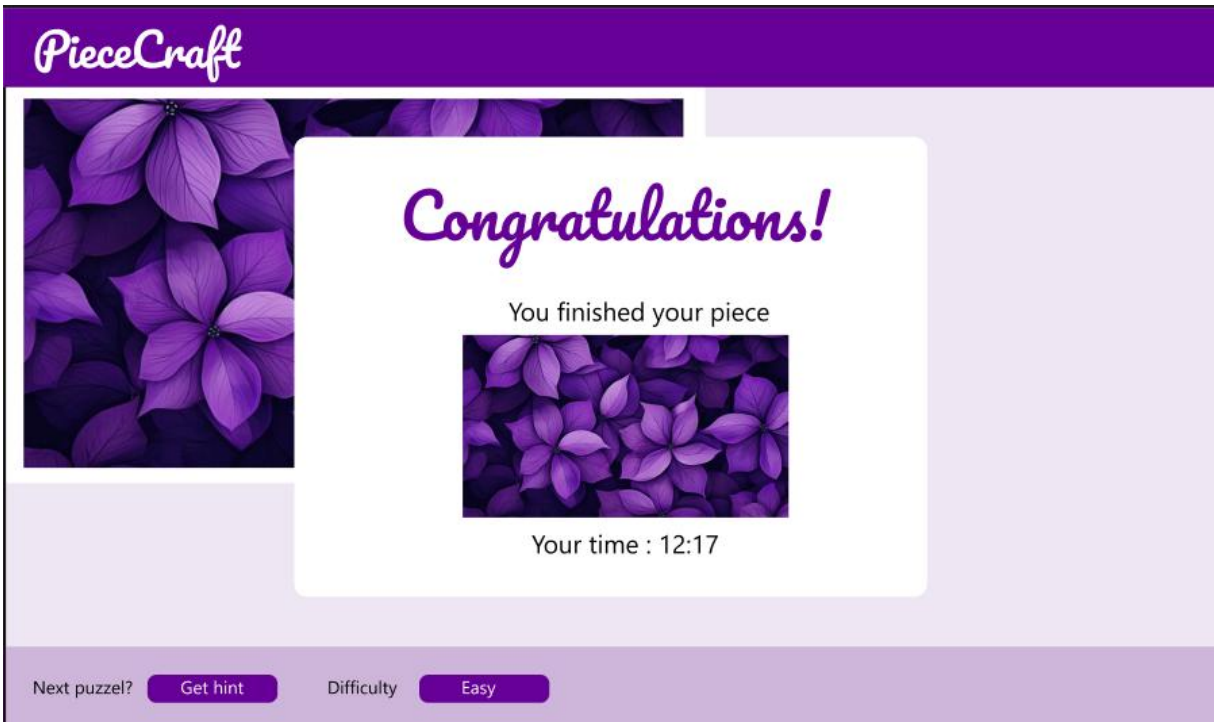
Technologia

Program napisany będzie w języku python.

GUI




PieceCraft
Next puzzel?
Get hint
Difficulty
Easy
Time: 2:31


PieceCraft
Congratulations!
You finished your piece
Your time : 12:17
Next puzzel?
Get hint
Difficulty
Easy