

Plan projektu

Projektem realizowanym przeze mnie będzie projekt nr. 12 o "rednim poziomie trudności" - kwadratowe ponumerowane puzzle, które należy ułożyć w kolejności rosnącej wierszami zgodnie z numerkami.

Wykorzystane narzędzia i biblioteki

GUI aplikacji zostanie napisane z użyciem frameworka Qt. Jeśli chodzi o IDE, to zostanie wykorzystany CLion. Tam gdzie będzie to konieczne, wykorzystany zostanie QtCreator. Kompilator: `clang`

Funkcjonalności

Tworzenie nowej gry

- Możliwość wyboru planszy spośród typów: numeryczne, obrazkowe
- Możliwość wyboru rozmiaru planszy spośród predefiniowanych rozmiarów, ale także możliwość zdefiniowania własnego rozmiaru planszy
- Możliwość wgrania własnego zdjęcia w celu zdefiniowania planszy z obrazkiem
- Możliwość zapisu planszy, w celu późniejszego szybszego rozpoczęcia gry
- Możliwość rozgrywki w trybie zwykłym, czyli zegar odlicza czas rozgrywki
- Możliwość rozgrywki w trybie walki z czasem - użytkownik ustawia limit czasu. Gra kończy się, gdy skończy się czas, lub gdy puzzle zostaną ułożone

Rozgrzywka

- Licznik odmierzający czas
- Licznik wykonanych ruchów
- Możliwość zrestartowania rozgrywki
- Sprawdzenie warunku zwycięstwa
- Możliwość wyciszenia podglądu planszy w celu sprawdzenia jak ma wyglądać wynikowa plansza

Koniec rozgrywki

- Wyświetlenie komunikatu pokazującego jak użytkownik wypadł na tle swoich poprzednich rozgrywek
- Wyświetlenie opcji "zagraj jeszcze raz"

Szkic GUI

Menu główne

Tworzenie nowej rozgrywki

Rozgrywka

Koniec gry

Diagramy klas

Ekran główny

Diagram zależności

Diagram klas

- ¥ **MainWindow** - jest to klasa-kontener, kt-ra zawiera logik! nawigacji opart& o stos. Klasa ta przechwytyje i obs&uguje sygna&y, kt-re wysy&aj& widoki-dzieci w celu nawigacji.
- ¥ **MenuScreen** - klasa reprezentuj&ca ekran g&-wny. W tym przypadku wyj&tkowo nie ma odpowiadaj&cego **ViewModel** u poniewa# ten ekran jest skrajnie prosty.

Ekran tworzenia nowej gry

Diagram zależności

Diagram klas

- ¥ **MainWindow** - klasa kontener działająca tak jak opisano w [tym akapicie](#).
- ¥ **GameSettingsScreen** - klasa reprezentująca ekran tworzenia nowej gry. Odpowiedzialnością tej klasy jest renderowanie ui oraz przechwytywanie interakcji użytkownika, która jest przekazywana do **GameSettingsScreenViewModel**. Klasa ta też jest odpowiedzialna za wysyłanie sygnałów do **MainWindow**, które oznaczają nawigację między ekranami.
- ¥ **BoardPreview** - customowy widget, który reprezentuje podgląd planszy podczas tworzenia nowej rozgrywki. Jego zadaniem jest pokazać użytkownikowi, jak będzie wyglądała plansza, z którą się zmierzy. Odpowiedzialnością tej klasy jest to, aby poprawnie wyświetlać plansz i poprawnie się skalować.
- ¥ **GameSettingsScreenViewModel** - odpowiada za utrzymywanie aktualnego stanu oraz wysyłanie sygnałów do **GameSettingsScreen** w przypadku, gdy ekran powinien się przerenderować na skutek zmiany stanu. Dodatkowo view model validuje czas wpisany w polu do ustawiania limitu.

Ekran gry

Diagram zależności dla ekranu gry

Diagram klas dla ekranu gry

- ¥ **GameScreen** - reprezentuje ekran gry. Odpowiada za poprawne renderowanie planszy, liczby ruchów, czasu; przechwytuje interakcje użytkownika i przekazuje je do **GameScreenViewModel**. Odpowiada za wysyłanie sygnałów do **MainScreen** w celu nawigowania. Wyświetla dialogi w przypadku wygranej/przegranej.
- ¥ **GameScreenViewModel** - odpowiada za zarządzanie stanem oraz obsługą interakcji użytkownika. Przechowuje stan planszy, zarządza licznikiem czasu, sprawdza warunek zwycięstwa, zleca wyświetlenie dialogów. Uaktualnia i porównuje najlepszy wynik dla konkretnego rozmiaru planszy.

- ¥ **RandomBoardFactory** - odpowiada za tworzenie losowej planszy dla podanego rozmiaru
- ¥ **Board** - klasa reprezentująca plansz! . Jest opisana szerzej w dalszej części dokumentu
- ¥ **GameWonDialog** - klasa reprezentująca dialog pojawiający się w momencie wygranej. Pokazuje użytkownikowi jego wynik w porównaniu do najlepszego.
- ¥ **GameLostDialog** - klasa reprezentująca dialog pojawiający się w trybie walki z czasem, gdy użytkownik nie zdąży rozwinąć planszy przed upływem czasu.
- ¥ **ResultStorage** - klasa pozwalająca na pobieranie i zapis najlepszych wyników dla konkretnego rozmiaru planszy.

- ¥ **Board** - klasa reprezentująca plansz!. Odpowiada za poprawne przestawianie elementów na planszy, pozwala na przywrócenie planszy do stanu początkowego. Przechowuje stan planszy.
- ¥ **Tile** - klasa reprezentująca abstrakcyjny typ puzzla. Zawiera pole oznaczające indeks kafelka w planszy.
- ¥ **ValueTile<T>** - klasa reprezentująca abstrakcyjny typ puzzla, który może poza indeksem przechowywać wartość, np. liczbę i literę itd.

- ¥ **NumberTile** - klasa reprezentująca kafelek, który zawiera wartość liczbową. Jest to kafelek wykorzystywany w klasycznej wersji gry, gdzie kafelki są ponumerowane od 1 do $n^2 - 1$, gdzie n to rozmiar planszy.
- ¥ **EmptyTile** - klasa reprezentująca pusty kafelek. Występuje na każdej planszy.