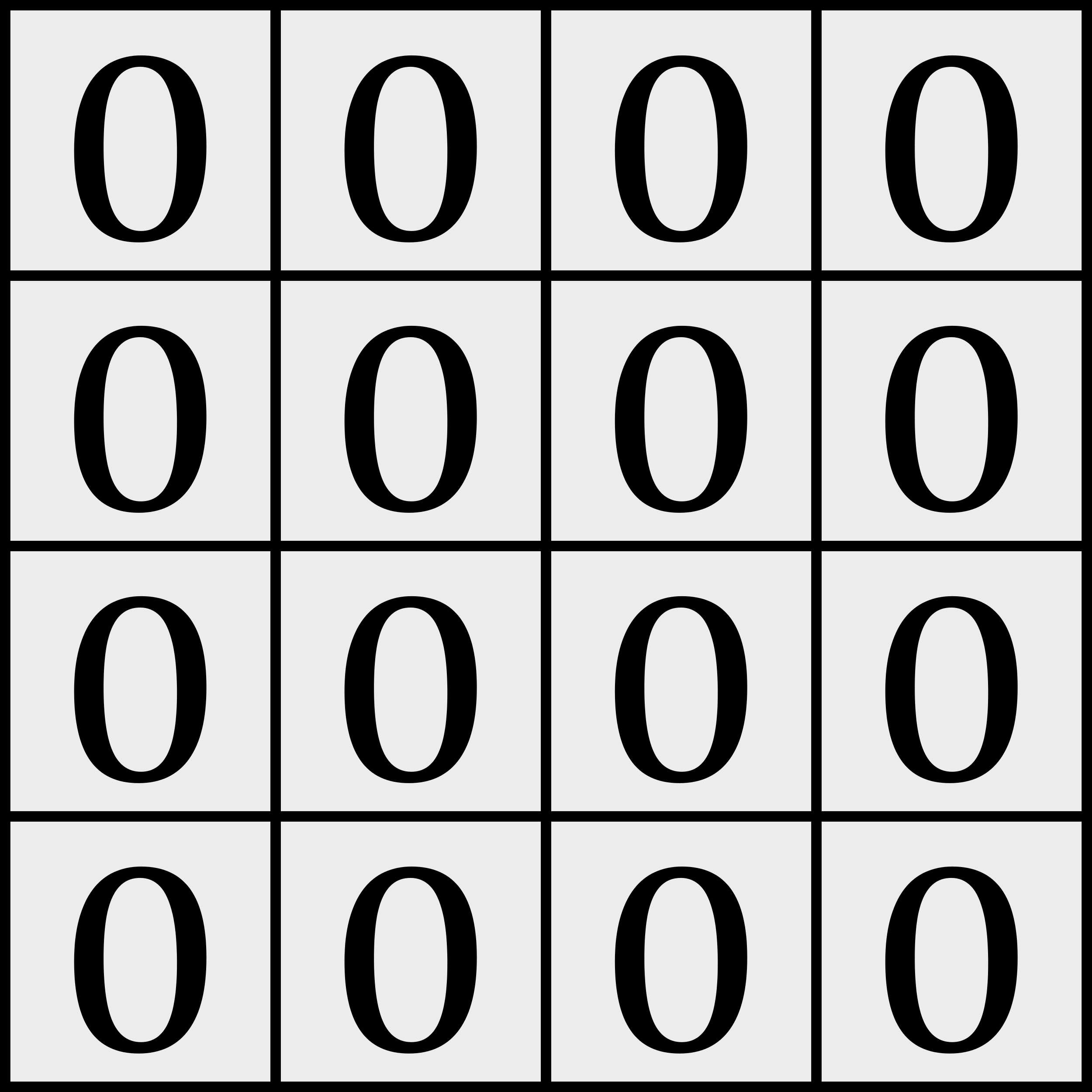
Dokumentacja

aplikacji Game2048

# Założenia projektowe

Wykorzystać QML do projektu aplikacji mulit-platformowej typu rozrywkowego. Projekty został opary na popularnej grze numerycznej 2048. Został wdrożony na systemy operacyjny Windows ( Windows 10 x64 ) oraz android ( arm64\_v8a ).

# Projekty Game2048

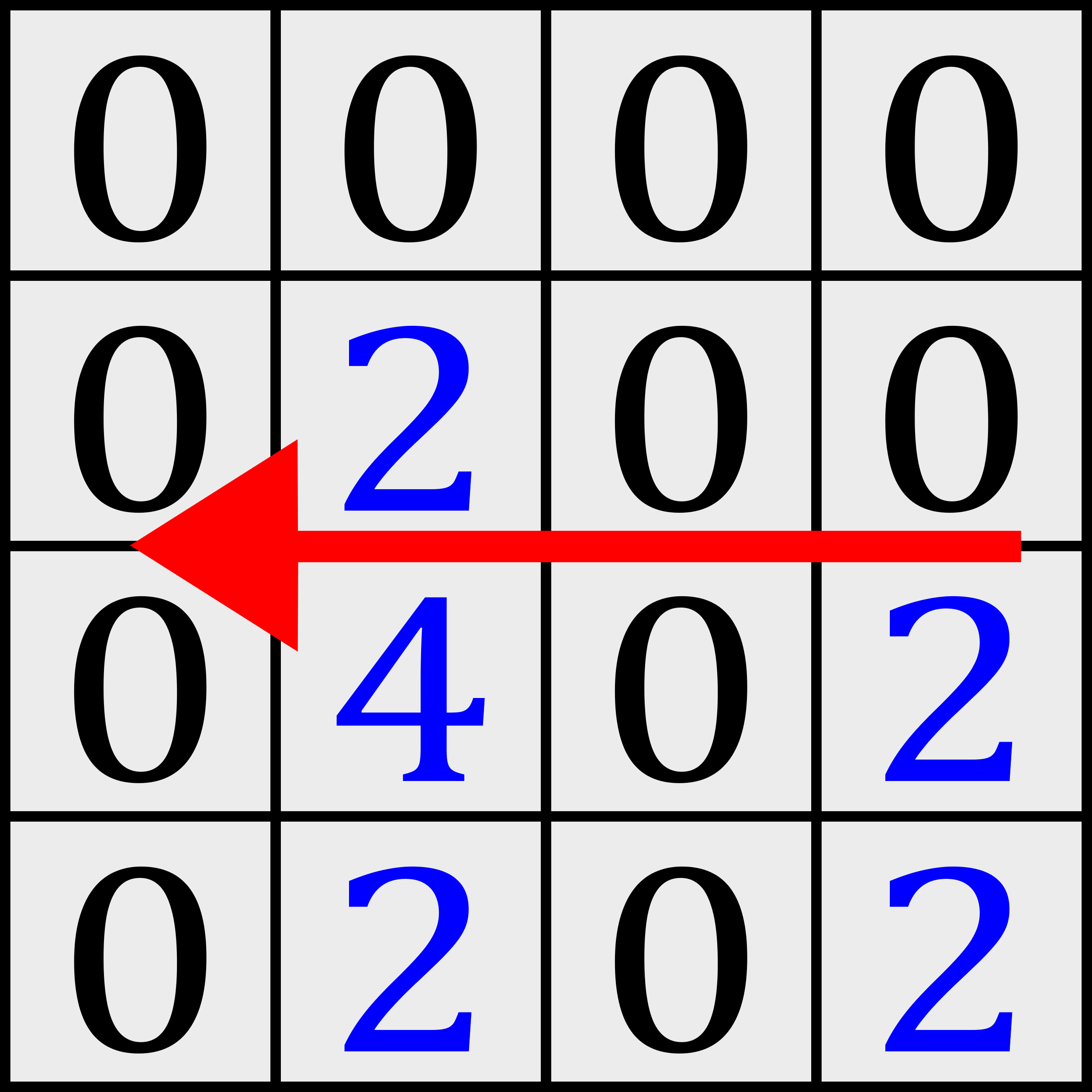
Projekty inspirowany [aplikacją 2048](https://play2048.co/).

Rysunek 1 Pole gry.

## Zasady gry

Gra polega na sumowaniu kolejnych potęg liczby 2 aż do osiągnięcia liczby 2048. Gra kończy się w momencie kiedy osiągnie się liczbę 2048 ( 211 ) lub kiedy zapełni się dostępne pole liczbami i nie będzie można już połączy w pary dwóch liczby.

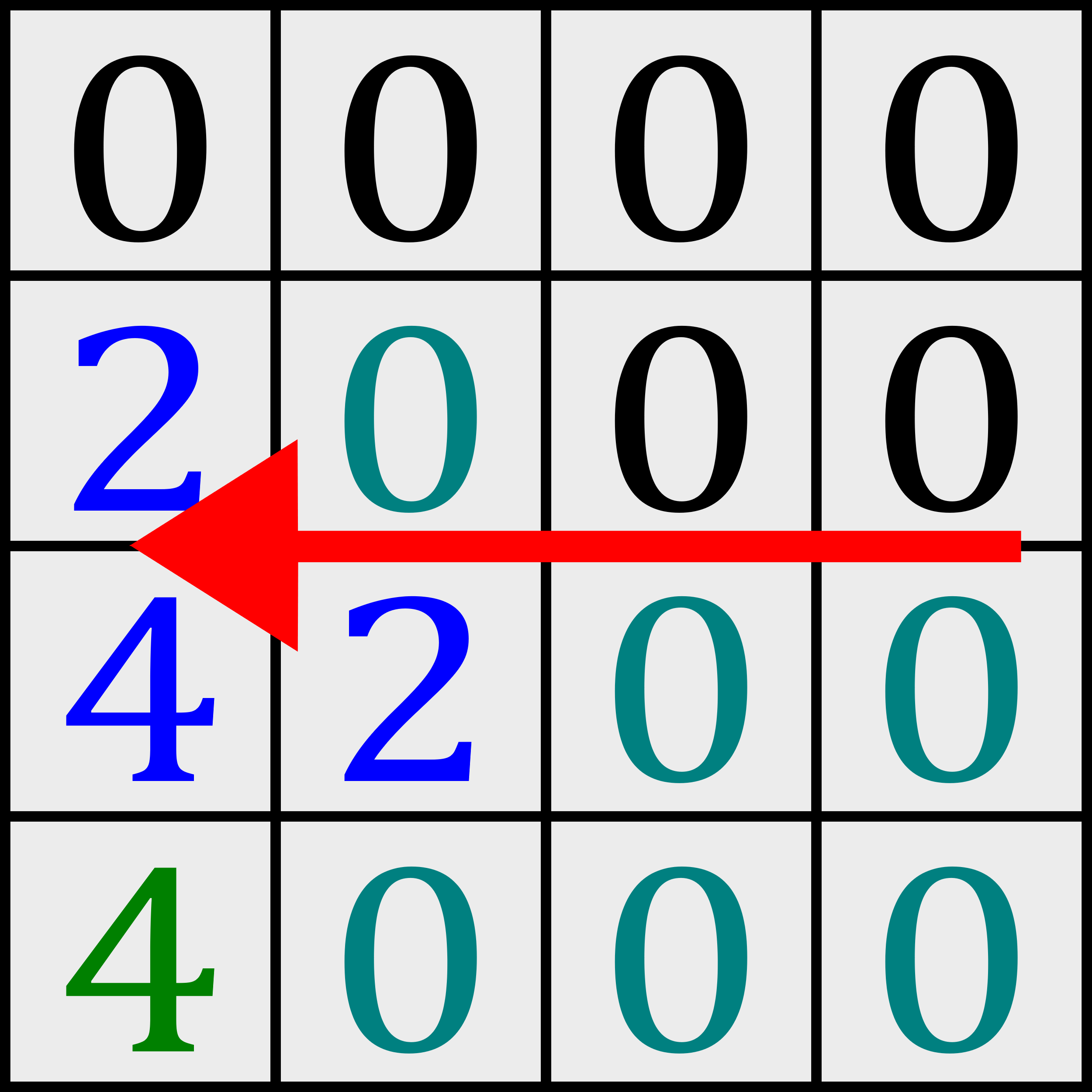
## Pole gry

Gra składa się z tablicy 4x4 ( Rysunek 1 ), która będzie wypełniana liczbami. Liczby w tablicy pojawia się losowo w miejscu gdzie jeszcze nie znajduje się liczba. Liczby które mogą być wylosowane to 2 i 4, a szansa na wylosowanie to dla 2: 9/10 i dla 4: 1/10. Liczby są dodawane po wykonaniu przez gracza ruchu oraz przy rozpoczęciu rozgrywki są losowane dwie liczby.

Rysunek 2 Przesunięcie w lewo  
( przed przesunięciem ).

## Przebieg rozgrywki

Graczy może przesuwać liczby w tablicy za pomocą czterech ruchów:

* Przesunięcie w lewo – wszystkie liczby w tablicy   
  ( oprócz zer ) są przesuwane w wierszu na lewą stronę zgodnie z poniższymi zasadami,
* Przesunięcie w prawo – wszystkie liczby w tablicy   
  ( oprócz zer ) są przesuwane w wierszu na prawą stronę zgodnie z poniższymi zasadami,
* Przesunięcie w górę – wszystkie liczby w tablicy   
  ( oprócz zer ) są przesuwane w kolumnie na górę   
  tablicy zgodnie z poniższymi zasadami,
* Przesunięcie w dół – wszystkie liczby w tablicy   
  ( oprócz zer ) są przesuwane w kolumnie na dół tablicy zgodnie z poniższymi zasadami.

Rysunek 3 Przesunięcie w lewo   
( po przesunięciu ).

## Zasady:

1. Przesunięć:
   1. Przesunięcie przerzucają wszystkie wartość oprócz zer na wybraną stronę po kolej. Tworząc w swego rodzaju stos elementów którego kierunek określa przesunięcie ( ruch gracza ) ( Rysunek 2 i Rysunek 3 ).
2. Sumowania:
   1. Liczby mogą się sumować raz na przesunięcie ( ruch gracza ),
   2. Liczby są sumowane jeżeli mają tą samą wartość i po między nimi nie znajduje się żadna inna liczba w   
      wierszu lub kolumnie podczas   
      przesunięcia ( ruch gracza) ( Rysunek 2 i Rysunek 3 ).

## Obliczanie wyniku punktacja

Za każde dodanie do siebie liczby gracz otrzymuje punkty w ilość sumy sumowanych wartość. Na przykłady sumowanie dwóch 2 daje 4 punkty, sumowanie dwóch 8 daje 16 punktów i tak dalej.

Można również rywalizować w grze na czas. Podczas rozgrywki jest mierzony czas podejścia i pod koniec jest widoczny na podsumowaniu.

# Realizacja projektu

Projekty składa się z następujących klas QML:

* Main QML Typ – Klas główna projektu tworząca główne okno programu i zarządzająca trybami pracy,
* FieldGame QML Typ – Klas tworząca pole gry i zarządzająca tym polem oraz zasadami rozgrywki,
* Cube QML Typ – Klas tworząca pojedynczą komórkę w polu gry,
* Klasy trybów / strony gry ( znajdują się w folderze Page ):
  + MenuPage QML Typ – Strona menu witająca gracza przy uruchomieniu aplikacji,
  + GamePage QML Typ – Strona reprezentująca wizualnie przebieg rozgrywki,
  + GameEndPage QML Typ – Strona wykorzystywana przez GamePage.qml do tworzenia informacji o zakończeniu rozgrywki.

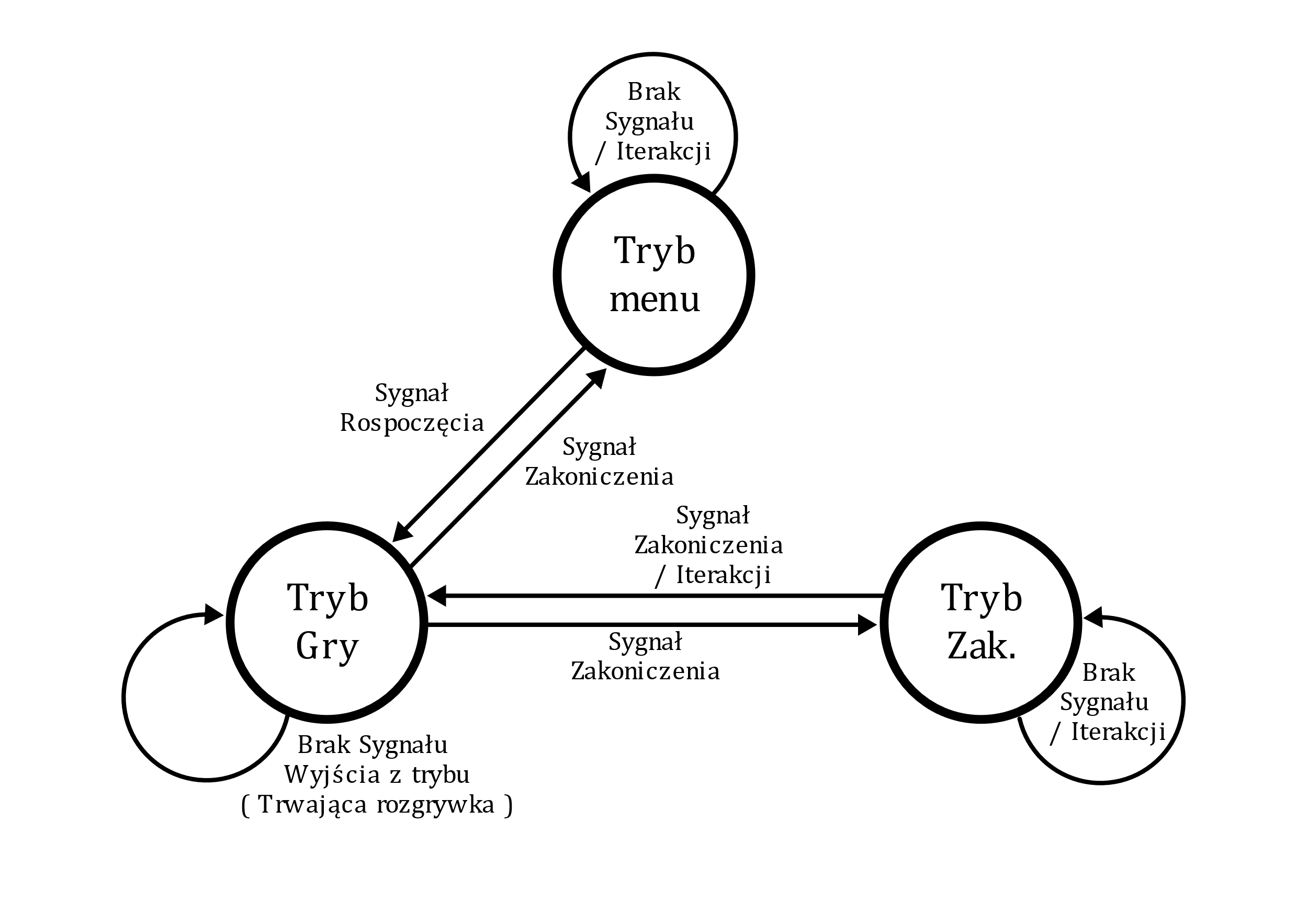
Wszystkie klasy korzystają z elementów biblioteki QtQuick 6.7.

Projekty zwiera również pliki main.cpp w który znajduje się funkcja main inicjalizująca silnik aplikacji qml.

## Tryb pracy ( maszyna stanów )

Aplikacja może znajdować się w trzech trybach:

* Tryb menu – gdzie możemy zobaczyć najlepszy wynik z poprzedniej rozgrywki oraz rozpocząć nową rozgrywkę,
* Tryb gry – gdzie odbywa się rozgrywka. Liczona jest punktacja i czas,
* Tryb zakończenia rozgrywki – wyświetla on podsumowanie rozgrywki oraz porwanie z najlepszy poprzednimi wynikiem.

Relacja pomiędzy trybami jest następująca:

Rysunek 4 Relacja Trybów pracy.

## Opis utworzonych klas i obiektów:

### Main QML Typ

Klas główna projektu tworząca główne okno programu i zarządzająca trybami pracy.

|  |  |
| --- | --- |
| Deklaracja importu | Klasa wewnętrzna projektu Game2048 |
| Dziedziczy | [Windows QML Typ](https://doc.qt.io/qt-6/qml-qtquick-window.html) |

#### Parapety

minPointSize: int  
maxPointSize: int  
fontfamily: string  
fontColor: string  
backgroundColor: string  
mainColor: string

#### Szczegółowy opis

Klas główna projektu tworząca główne okno programu ( QML Windows ) i zarządzająca trybami pracy. Przełącza trybu pracy po między obiektami: MenuPage oraz GamePage, które reprezentuje wizualną interpretacje tych trupów.

#### Dokumentacja Parametrów

##### minPointSize: int

Parametry tylko do odczytu. Przechowuje minimalną wielkość tekstu dla aplikacji.

##### maxPointSize: int

Parametry tylko do odczytu. Przechowuje maksymalną wielkość tekstu dla aplikacji.

##### fontfamily: string

Parametry tylko do odczytu. Przechowuje nazwę czcionki tekstu dla aplikacji.

##### fontColor: string

Parametry tylko do odczytu. Przechowuje kolor tekstu dla aplikacji.

##### backgroundColor: string

Parametry tylko do odczytu. Przechowuje kolor tła dla aplikacji.

##### mainColor: string

Parametry tylko do odczytu. Przechowuje główny kolor „motywu” dla aplikacji.

### FieldGame QML Typ

Klas tworząca pole gry i zarządzająca tym polem oraz zasadami rozgrywki.

|  |  |
| --- | --- |
| Deklaracja importu | Klasa wewnętrzna projektu Game2048 |
| Dziedziczy | [Item QML Typ](https://doc.qt.io/qt-6/qml-qtquick-item.html) |

#### Parapety

tabGameData: var  
tabGameDataSum: var  
scoresGame: int  
minPointSize: int  
maxPointSize: int  
fontfamily: string  
fontColor: string  
backgroundColor: string  
mainColor: string  
borderColor: string

#### Sygnały

changeTabGameData()  
changeScoresGame()  
newGame()  
gameOver()  
gameWin()

#### Metody

shiftLeft()  
shiftRight()  
shiftUp()  
shiftDown()  
addNewNumebr()  
clearTabGameData()  
clearTabGameDataSum()  
isNumberInTabGameData(number: var): bool

#### Szczegółowy opis

Klas tworząca pole gry i zarządzająca tym polem oraz zasadami rozgrywki. Pole gry składa się z 16 obiektów Cube ułożonych w macierz 4x4.

Analizuje zdarzenia wejściowe otrzymane od użytkownika takie jak: kliknięcie przycisku na klawiaturze, przesunięcie myszą lub palcem ( gesty mysz lub palca ).

* A / left arrow / gest lewo mysz / palce – przesunięcie w lewo ( Metoda shuftLeft()),
* D / right arrow / gest prawo mysz / palce – przesunięcie w prawo ( Metoda shuftLeft()),
* W / up arrow / gest do góry mysz / palce – przesunięcie do góry ( Metoda shuftUP()),
* S / down arrow / gest w dół mysz / palce – przesunięcie w dół ( Metoda shuftDown()).

Po otrzymaniu zdarzeni wejściowych od użytkownika wykonuje odpowiedni metody odpowiadające za zasady rozgrywki.

Odbiera i wysłał odpowiednie sygnał takie jak: newGame(), gameOver(), gameWin().

#### Dokumentacja Parametrów

##### tabGameData: var

Domyślna wartość 16x[0]. Tablica 16 elementowa przechowująca aktualne wartość pola gry.

##### tabGameDataSum: var

Domyślna wartość 16x[0]. Tablica 16 elementowa przechowująca wartość zaistnienia operacji sumy pomiędzy komórkami tablicy tabGameData. Podczas operacji: shiftLeft(), shiftRight(), shiftUp(), shiftDown() może dojść do sumowania dwóch wartość z dwóch komórek zgodnie z regułami gry. Uzyskana tak wartość nie może zostać ponownie sumowana w ty samym wywołaniu powyższych metod, stąd powstanie pomocniczej tablicy.

##### scoresGame: int

Domyślna wartość 0. Parametry przechowując wynik aktualnej rozgrywki.

##### minPointSize: int

Domyślna wartość 12. Parametry powiązany z Main.minPointSize: int. Przechowuje minimalną wielkość tekstu dla obiektu.

##### maxPointSize: int

Domyślna wartość 200. Parametry powiązany z Main.maxPointSize: int. Przechowuje maksymalną wielkość tekstu dla obiektu.

##### fontfamily: string

Domyślna wartość „Sagoe UI”. Parametry powiązany z Main.fontfamily: string. Przechowuje nazwę czcionki tekstu dla obiektu.

##### fontColor: string

Domyślna wartość „#000000”. Parametry powiązany z Main.fontColor: string. Przechowuje kolor tekstu dla obiektu.

##### backgroundColor: string

Domyślna wartość „#000000”. Parametry powiązany z Main.backgroundColor: string. Przechowuje kolor tła dla obiektu.

##### mainColor: string

Domyślna wartość „#ffffff”. Parametry powiązany z Main.mainColor: string. Przechowuje główny kolor „motywu” dla obiektu.

##### borderColor: string

Domyślna wartość „#000000”. Przechowuje kolor ramki na przykład obiektu Cube.

#### Dokumentacja Sygnałów

##### changeTabGameData()

Sygnał emitowana za każdym razem gry któraś z metod zmodyfikuje  
tablice tabGameData: var. Sygnał ten przesyłany jest do utworzonych 16 obiektów Cube.

##### changeScoresGame()

Sygnał emitowana za każdym razem gry któraś z metod zmodyfikuje parametry scoresGame: int.

##### newGame()

Sygnał służący do ustawienia warunków początkowych dla nowej gry. Powinien zostać edytowany przez utworzony obiekty. Uwag nie jest on emitowany przez żadną metodę klas.

##### gameOver()

Sygnał emitowana gdy metoda addNewNumebr() wykryje że tablica jest zapełniona. Związane jest to z zasadą zakończenia rozgrywki z warunkiem „przegrana”.

##### gameWin()

Sygnał emitowana gdy metody shiftLeft(), shiftRight(), shiftUp() i shiftDown() wykryją że w tablicy powstała liczba 2048. Związane jest to z zasadą zakończenia rozgrywki z warunkiem „wygrana”.

#### Dokumentacja Metody

##### shiftLeft() shiftRight() shiftUp() shiftDown()

Metody przekształcające tablice tabGameData: var zgodnie z zasadami rozgrywki.

Przesunięcie przerzucają wszystkie wartość oprócz zer na wybraną stronę po kolej. Tworząc w swego rodzaju stos elementów którego kierunek określa przesunięcie.

Metody również podczas zdarzenia sumowania dwóch identycznych liczby dokonuje powiększenia wyniku rozgrywki.

##### addNewNumebr()

Metody dodająca nową wartość do tablicy tabGameData: var zgodnie z zasadami rozgrywki.

Na początku jest losowane jedno z pul tablicy które jest równe zero, a następnie wartość dodawanej liczby. Liczby które mogą być wylosowane to 2 i 4, a szansa na wylosowanie to dla 2: 9/10 i dla 4: 1/10.

Gdy nie ma wolnych pul w tablicy tabGameData: var emitowany jest sygnał gameOver(), który jest równoważny końcowy rozgrywki z warunkiem „przegrana”.

Metoda jest wywoływana dwa razy przez sygnał newGame().

##### clearTabGameData()

Metody czyści zawartość ( ustawia zera ) tablicy tabGameData: var. Metoda jest wywoływana przez sygnał newGame().

##### clearTabGameDataSum()

Metody czyści zawartość ( ustawia zera ) tablicy tabGameDataSum: var. Metoda jest wywoływana pod koniec każdej metody: shiftLeft(), shiftRight(), shiftUp() i shiftDown().

##### isNumberInTabGameData(number: var): bool

Metody sprawdza czy w tabeli tabGameData: var znajduje się podana liczba. Metoda jest wykorzystywana do sprawdzenia warunku zwycięstwa w grze ( czy jest 2048 ) każdy razowo po przesunięciu wartość w tablicy za pomocą metod shiftLeft().

Number: var – liczba którą chcemy wyszukać w tablicy.

return: bool – kiedy liczby jest w tablic zwraca true jeżeli nie zwraca false.

### Cube QML Typ

Klas tworząca pojedynczą komórkę w polu gry.

|  |  |
| --- | --- |
| Deklaracja importu | Klasa wewnętrzna projektu Game2048 |
| Dziedziczy | [Rectangle QML Typ](https://doc.qt.io/qt-6/qml-qtquick-rectangle.html) |

#### Parapety

minPointSize: int  
maxPointSize: int  
fontfamily: string  
fontColor: string  
text: string

#### Obraz zawierający Grafika, żółty, zrzut ekranu, krąg Opis wygenerowany automatycznieSzczegółowy opis

Klas tworząca pojedynczą komórkę w polu gry. Komórka składa się z głównego Rectangla QML oraz Text QML.

Cube ma kształty kwadratu i można w nim modyfikować rozmiary, teksty, kolor tła, kolor ramki oraz wielkość ramki.

Rysunek 5 Cube

#### Dokumentacja Parametrów

##### minPointSize: int

Domyślna wartość 12. Parametry powiązany z Main.minPointSize: int. Przechowuje minimalną wielkość tekstu dla obiektu.

##### maxPointSize: int

Domyślna wartość 200. Parametry powiązany z Main.maxPointSize: int. Przechowuje maksymalną wielkość tekstu dla obiektu.

##### fontfamily: string

Domyślna wartość „Sagoe UI”. Parametry powiązany z Main.fontfamily: string. Przechowuje nazwę czcionki tekstu dla obiektu.

##### fontColor: string

Domyślna wartość „#000000”. Parametry powiązany z Main.fontColor: string. Przechowuje kolor tekstu dla obiektu.

##### text: string

Domyślna wartość „”. Przechowuje wartość wyświetlanego tekstu przez  
obiekty Text QML.

#### Domyślne wartość parametrów dla Rectangle QML:

color: "#ffffff" Domyślny kolor tła,  
border.color: "#000000" Domyślny kolor ramki,  
border.width: 10 Domyślna wielkość ramki.

### MenuPage QML Typ

Strona menu witająca gracza przy uruchomieniu aplikacji.

|  |  |
| --- | --- |
| Deklaracja importu | Klasa wewnętrzna projektu Game2048 |
| Dziedziczy | [Item QML Typ](https://doc.qt.io/qt-6/qml-qtquick-item.html) |

#### Parapety

bestScores: int  
bestTime: string  
minPointSize: int  
maxPointSize: int  
fontfamily: string  
fontColor: string  
backgroundColor: string  
mainColor: string

#### Obraz zawierający tekst, zrzut ekranu, Czcionka, Grafika Opis wygenerowany automatycznieSygnały

Rysunek 6 Menu Page

startGame()

#### Szczegółowy opis

Klas tworzy wizualną stronę menu witającą użytkownika uruchamiającego aplikacje.

Strona zawiera Text QML: zawierającą nazwę gry, napis start oraz najlepszy wynik rozgrywki w tej sesji.

Aby uruchomić rozgrywkę należy dotknąć ekranu kliknąć muszką lub nacisnąć spacje.

#### Dokumentacja Parametrów

##### bestScores: int

Domyślna wartość 0. Parametry przechowuje najlepszy wyniki rozgrywki w tej sesji.

##### bestTime: string

Domyślna wartość „00:00:00”. Parametry przechowuje czas dla najlepszego wyniku w tej sesji.

##### minPointSize: int

Domyślna wartość 12. Parametry powiązany z Main.minPointSize: int. Przechowuje minimalną wielkość tekstu dla obiektu.

##### maxPointSize: int

Domyślna wartość 200. Parametry powiązany z Main.maxPointSize: int. Przechowuje maksymalną wielkość tekstu dla obiektu.

##### fontfamily: string

Domyślna wartość „Sagoe UI”. Parametry powiązany z Main.fontfamily: string. Przechowuje nazwę czcionki tekstu dla obiektu.

##### fontColor: string

Domyślna wartość „#000000”. Parametry powiązany z Main.fontColor: string. Przechowuje kolor tekstu dla obiektu.

##### backgroundColor: string

Domyślna wartość „#000000”. Parametry powiązany z Main.backgroundColor: string. Przechowuje kolor tła dla obiektu.

##### mainColor: string

Domyślna wartość „#ffffff”. Parametry powiązany z Main.mainColor: string. Przechowuje główny kolor „motywu” dla obiektu.

#### Dokumentacja Sygnałów

##### startGame()

Sygnał służący do inicjalizacji nowej gry oraz ustawienia warunków początkowych dla nowej gry. Powinien zostać edytowany przez utworzony obiekty. Uwag nie jest on emitowany przez żadną metodę klas.

### GamePage QML Typ

Strona reprezentująca wizualnie przebieg rozgrywki.

|  |  |
| --- | --- |
| Deklaracja importu | Klasa wewnętrzna projektu Game2048 |
| Dziedziczy | [Item QML Typ](https://doc.qt.io/qt-6/qml-qtquick-item.html) |

#### Parapety

bestScoresGame: int  
scoresGame: int  
bestTimeGame: string  
timeGame: string  
minPointSize: int  
maxPointSize: int  
fontfamily: string  
fontColor: string  
backgroundColor: string  
mainColor: string

#### Sygnały

Rysunek 7 Game Page

endGame()

#### Szczegółowy opis

Klas tworzy wizualną stronę rozgrywki w której będzie odbywać się cała wizualna część rozgrywki.

Strona zawiera: trzy Cube QML do prezentacji wyników i zmierzonego czasu, przycisk do rozpoczynania nowej rozgrywki na bazie Cube QML, FieldGame QML pole z rozgrywką oraz zarządza stroną GameEndPage

Przycisk New Game reaguje na kliknięcie muszą, dotknięcie palce. Nową rozgrywkę można również rozpocząć naciskając spacje na klawiaturze co jest obsługiwane przez FieldGame QML.

#### Dokumentacja Parametrów

##### bestScoresGame: int

Domyślna wartość 0. Parametry przechowuje najlepszy wyniki rozgrywki w tej sesji.

##### scoresGame: int

Domyślna wartość 0. Parametry przechowuje aktualny wyniki rozgrywki.

##### bestTimeGame: string

Domyślna wartość „00:00:00”. Parametry przechowuje czas dla najlepszego wyniku w tej sesji.

##### timeGame: string

Domyślna wartość „00:00:00”. Parametry przechowuje czas dla aktualnej rozgrywki.

##### minPointSize: int

Domyślna wartość 12. Parametry powiązany z Main.minPointSize: int. Przechowuje minimalną wielkość tekstu dla obiektu.

##### maxPointSize: int

Domyślna wartość 200. Parametry powiązany z Main.maxPointSize: int. Przechowuje maksymalną wielkość tekstu dla obiektu.

##### fontfamily: string

Domyślna wartość „Sagoe UI”. Parametry powiązany z Main.fontfamily: string. Przechowuje nazwę czcionki tekstu dla obiektu.

##### fontColor: string

Domyślna wartość „#000000”. Parametry powiązany z Main.fontColor: string. Przechowuje kolor tekstu dla obiektu.

##### backgroundColor: string

Domyślna wartość „#000000”. Parametry powiązany z Main.backgroundColor: string. Przechowuje kolor tła dla obiektu.

##### mainColor: string

Domyślna wartość „#ffffff”. Parametry powiązany z Main.mainColor: string. Przechowuje główny kolor „motywu” dla obiektu.

#### Dokumentacja Sygnałów

##### endGame()

Sygnał służący do inicjalizacji zakończenia rozgrywki. Emitowany jest przez obiekty GameEndPage QML pod naciśnięciu / dotknięciu ekranu. Powinien zostać edytowany przez utworzony obiekty.

### GameEndPage QML Typ

Strona wykorzystywana przez GamePage.qml do tworzenia informacji o zakończeniu rozgrywki.

|  |  |
| --- | --- |
| Deklaracja importu | Klasa wewnętrzna projektu Game2048 |
| Dziedziczy | [Item QML Typ](https://doc.qt.io/qt-6/qml-qtquick-item.html) |

#### Parapety

#### Sygnały

Rysunek 8 Game End Page

#### Szczegółowy opis

Klas tworzy wizualną stronę zakończenia rozgrywki w której została podsumowana obecna rozgrywka i porównana do najlepszej rozgrywki w tej sesji.

Strona zawiera Text QML: zwierający informacje o przegranej lub wygranej, wynik i czas obecnej rozgrywki oraz wynik i czas najlepszej rozgrywki w tej sesji.

Aby zaakceptować informacje i wrócić do menu należy dotknąć ekranu kliknąć muszką lub nacisnąć spacje.

#### Dokumentacja Parametrów

##### bestScoresGame: int

Domyślna wartość 0. Parametry przechowuje najlepszy wyniki rozgrywki w tej sesji.

##### scoresGame: int

Domyślna wartość 0. Parametry przechowuje aktualny wyniki rozgrywki.

##### bestTimeGame: string

Domyślna wartość „00:00:00”. Parametry przechowuje czas dla najlepszego wyniku w tej sesji.

##### timeGame: string

Domyślna wartość „00:00:00”. Parametry przechowuje czas dla aktualnej rozgrywki.

##### TextGameEnd: string

Domyślna wartość „”. Parametry przechowuje informacje o zwycięstwie lub przegranej.

##### minPointSize: int

Domyślna wartość 12. Parametry powiązany z Main.minPointSize: int. Przechowuje minimalną wielkość tekstu dla obiektu.

##### maxPointSize: int

Domyślna wartość 200. Parametry powiązany z Main.maxPointSize: int. Przechowuje maksymalną wielkość tekstu dla obiektu.

##### fontfamily: string

Domyślna wartość „Sagoe UI”. Parametry powiązany z Main.fontfamily: string. Przechowuje nazwę czcionki tekstu dla obiektu.

##### fontColor: string

Domyślna wartość „#000000”. Parametry powiązany z Main.fontColor: string. Przechowuje kolor tekstu dla obiektu.

##### backgroundColor: string

Domyślna wartość „#ffffff”. Przechowuje kolor tła dla obiektu.

#### Dokumentacja Sygnałów

##### clicked()

Sygnał jest emitowany kiedy użytkowni wejdzie w interakcje ze stroną poprzez kliknięcie myszą, dotknięcie palce lub naciśnięcie na klawiaturze spacji.