Politechnika
Świętokrzyska
2017
Informatyka
Semestr III

Wychowski Norbert

Zachariasz Łukasz

Projekt z przedmiotu

Programowanie
w języku C 2

Temat projektu

Gra Snake

Opis tematyki projektu

Tematem projektu jest wydana po raz pierwszy w połowie lat siedemdziesiątych XX wieku gra komputerowa „Snake". Jest to zręcznościowa gra, w której grający kontroluje kierunek w którym porusza się stworzenie zwane tytułowym Wężem. Gracz ma za zadanie tak sterować głównym bohaterem aby zbierać jedzenie pojawiające się na planszy (lub inne przedmioty w zależności od zamysłu twórcy), próbując nie uderzyć głową węża w krawędzie planszy, jego tułów lub przeszkody przez niego pozostawione.

Zebrany pokarm to oczywiście punkty dla gracza, ma on ich zdobyć jak najwięcej, jednak najistotniejszym aspektem jest zwiększanie się długości węża. Wąż robi się coraz dłuższy co naturalnie utrudnia graczowi zadanie. Wymaga to coraz większej uwagi i czasami strategicznego podejścia do obieranej trasy przez węża do kolejnego przedmiotu.

Sterowanie jest najprostszym z możliwych najczęściej służą do tego cztery klawisze kierunkowe.

Użyte technologie

Projekt został napisany obiektowo w języku programowania C++ w oparciu o Framework QT, który jest zestawem przenośnych bibliotek i narzędzi programistycznych dedykowanych m.in. dla języków C++, QML i Java. Zawiera m.in. Klasy służące do budowy graficznego interfejsu programów, obsługi plików, również multimedialnych takich jak grafiki czy dźwięki.

Środowisko QT obsługuje wiele platform m.in. Mac OS X, Android, Linuks.

Nie ogranicza do tworzenia programów pod konkretny system, jest wykorzystywany przez znane na całym świecie firmy programistyczne, posiada bardzo bogatą dokumentację i wsparcie, stąd też jego wybór.

Kompilacja wykonywana była przy użyciu MS Visual C++ 2013 jako jednym z wielu współpracujących ze środowiskiem QT.

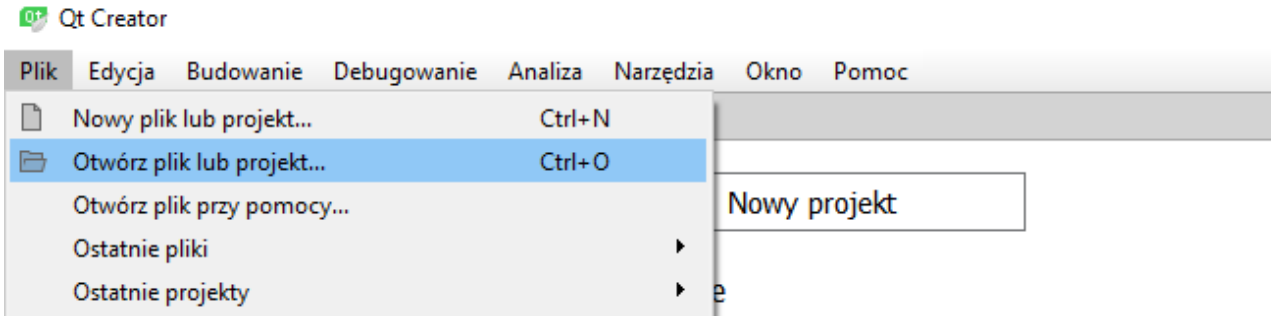
Projekt został stworzony z myślą o kompilacji i uruchomieniu w systemie Windows.

Instrukcja kompilacji i uruchomienia

Do kompilacji projektu potrzebny jest oczywiście Framework QT oraz w.w. kompilator. Instalacja QT jest bardzo prosta i nie wymaga dłuższego opisu, wystarczy pobrać odpowiednią wersję pod odpowiedni kompilator. QT automatycznie wykrywa zainstalowane kompilatory więc nie ma tutaj potrzeby przeprowadzania znanej skądinąd irytującej konfiguracji środowiska.

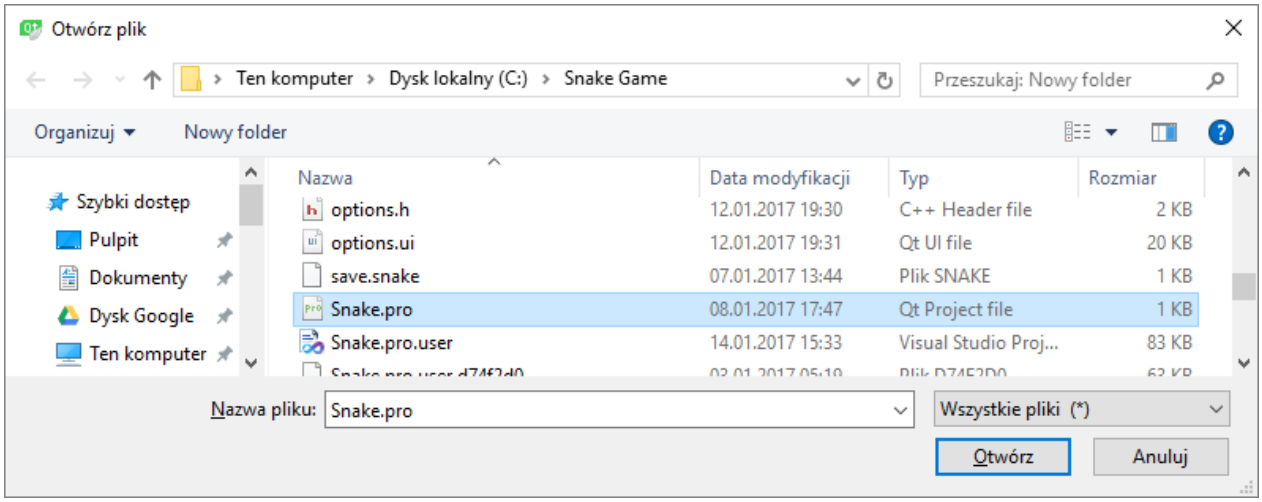
Już na samym początku warto zaznaczyć, że ścieżka do folderu z plikami projektu nie może zawierać polskich znaków diakrytycznych. Spowoduje to błąd kompilacji, niestety nie informując o przyczynie.

Mając folder z plikami projektu, możemy przystąpić do jego otwarcia. Zaczynamy od uruchomienia QT Creatora. Następnie w głównym oknie z menu Plik wybieramy Otwórz plik lub projekt...

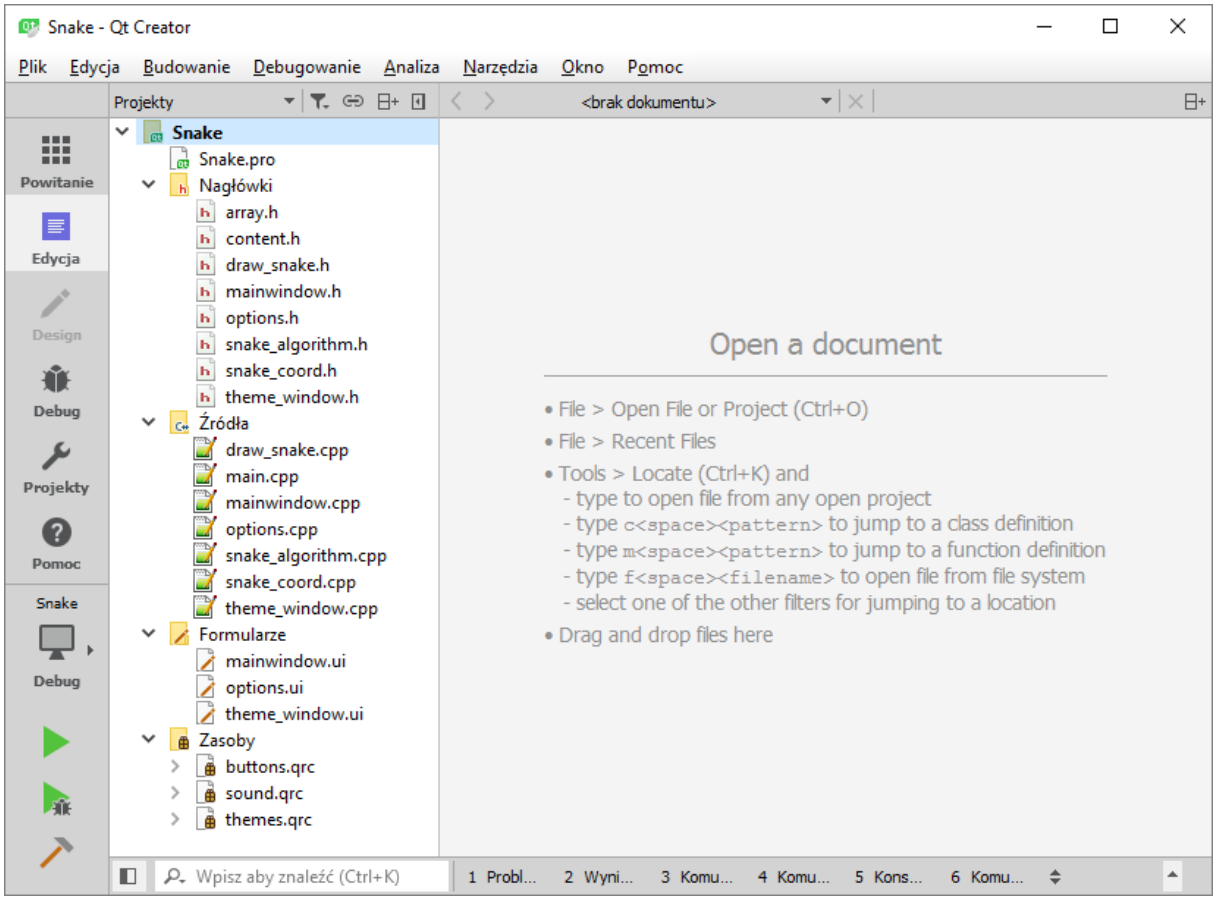


Następnie udajemy się do naszego folderu z projektem i wskazujemy plik z rozszerzeniem *.pro

W tym konkretnym projekcie jest to Snake.pro.



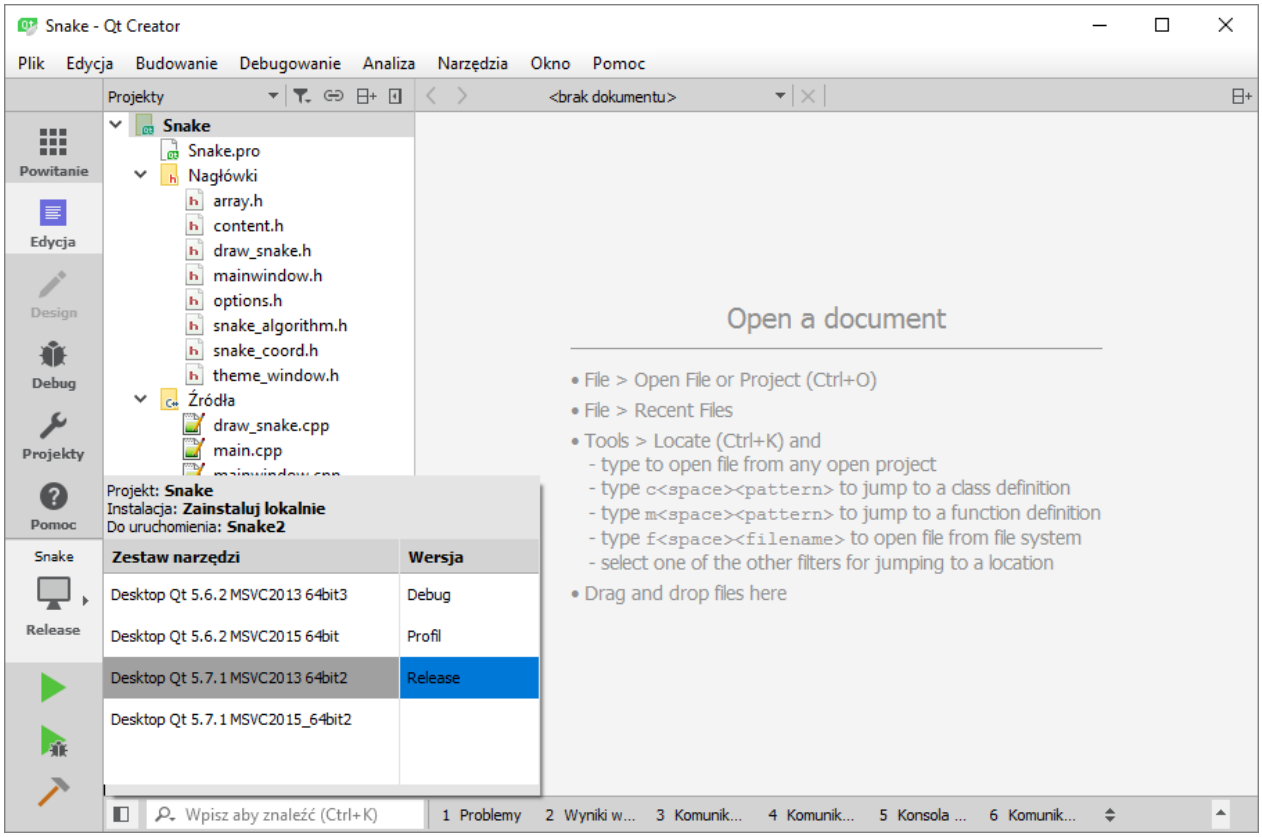
Po chwili zobaczymy wczytaną strukturę projektu.



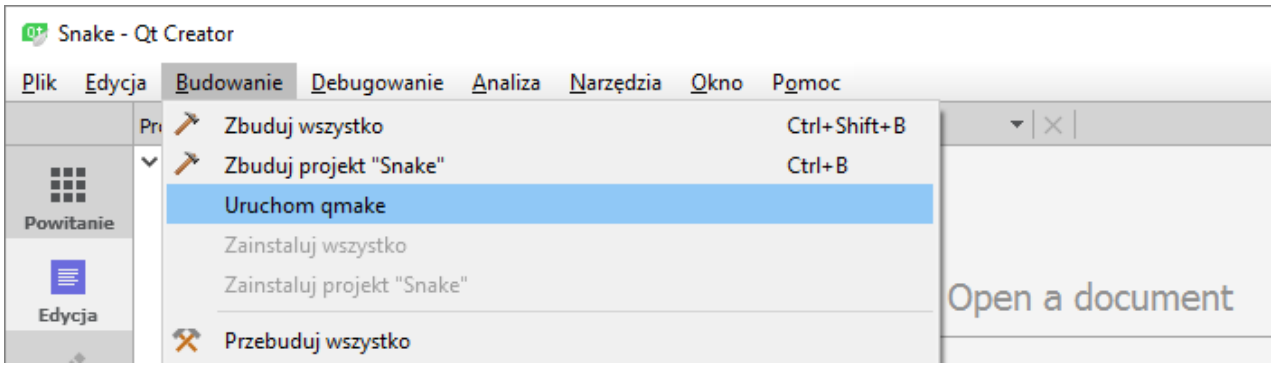
Wybieramy odpowiedni rodzaj narzędzi,

czyli wersję QT jeśli pobraliśmy więcej niż jedną,

oraz wersję kompilacji. Projekt jest gotowy do kompilacji.



Wybieramy Budowanie i Uruchom qmake.



Następnie, jak wyżej, wybieramy Zbuduj projekt "Snake".

Pozostało już tylko uruchomić grę, można użyć do tego skrótu klawiszowego Ctrl + R. Po chwili zobaczymy główne okno gry.

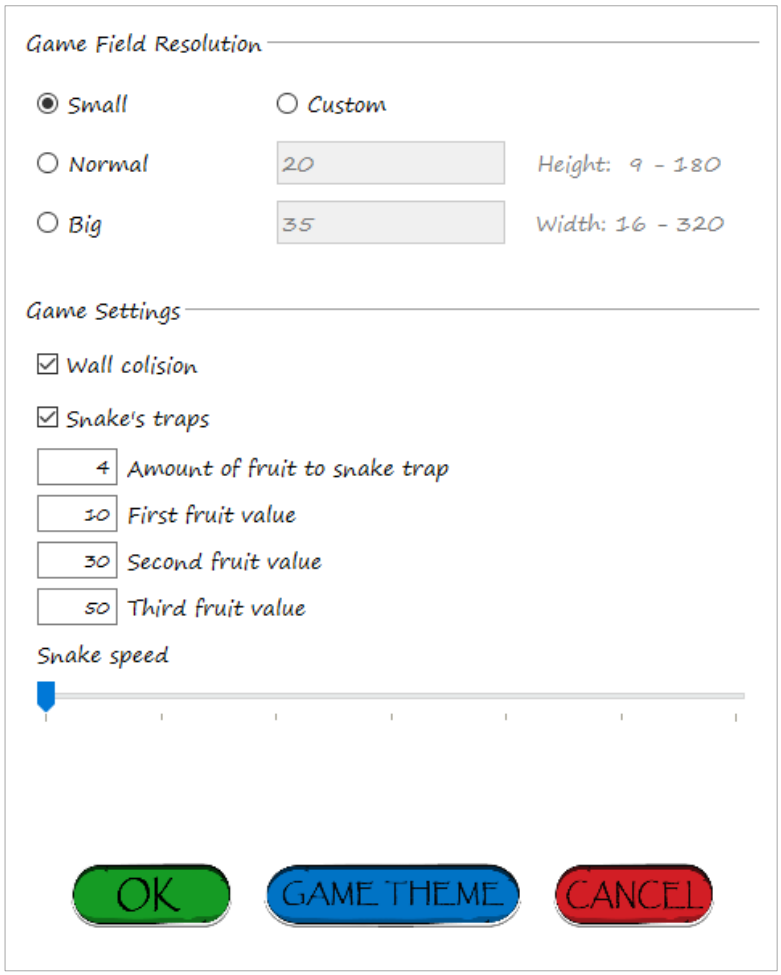
Roli przycisków raczej nikomu wyjaśniać nie trzeba, ciekawostką, której na pierwszy rzut oka nie widać jest możliwość przesuwania okien gry przez „chwytanie" ich myszą w dowolnym miejscu z pominięciem elementów interaktywnych interfejsu (np. przyciski).

Nie bez powodu mowa o oknach bo jest ich więcej. Zanim zaczniemy rozgrywkę warto sprawdzić jak możemy ją dostosować.

Klikając Options otwiera się kolejne okno w którym możemy ustawić:

- Rozdzielczość planszy gry – im mniejsza tym elementy w grze będą większe.
- Kolizje węża ze ścianą.
- Ilość owoców których zjedzenie generuje pułapkę.
- Ilość punktów dla zebranego typu owocu.
- Prędkość poruszania się węża.

Okno opcji prezentuje się w następująco.



Ostatnia opcja ukryta jest pod przyciskiem Game Theme.

Zostaje wywołane okno prezentujące wybór stylu graficznego gry. Do wyboru mamy cztery warianty, które zmieniają wygląd węża, podłoża, po którym się porusza, pułapek i zbieranego pokarmu.

Wystarczy kliknąć wybrany motyw i potwierdzić bądź nie, przyciskiem OK lub Cancel. Analogicznie w opcjach.

Gdy już znane są możliwości personalizacji rozgrywki, warto do niej przejść. Po kliknięciu na przycisk Start ukazuje nam się pole gry, wąż jest w ruchu, sterowanie realizowane jest przy użyciu strzałek kierunkowych bądź klawiszy W,S,A,D.

W tle odtwarzana jest muzyka oraz dźwięki łapania owoców. W dolnej części okna widoczna jest obecna ilość punktów, najlepszy wynik oraz czas rozgrywki.

W dowolnym momencie gdy gra się toczy, możemy ją zatrzymać przyciskiem Spacji lub klawiszem Escape. Ukaże się ekran główny lecz dodatkowo widoczna będzie informacja o możliwości kontynuowania gry.

Dwukrotne wciśnięcie Escape podczas gry zamyka ją.

W momencie gdy nastąpi przegrana jesteśmy informowani dźwiękiem, muzyka grana w tle zostaje zatrzymana a po chwili gra wyświetla następujący ekran.

Rola przycisków i w tym przypadku nie powinna mieć wątpliwości. Możemy od razu ponowić grę lub wrócić do menu jeśli chcemy zmienić pewne rzeczy w opcjach.

Dodatkowo widzimy zestawienie wyniku punktowego ostatniej rozegranej gry z najlepszym jaki zdobyto.

Jeśli chodzi o rozgrywkę to pozostaje już tylko życzyć pobijania coraz to większych rekordów.

Najważniejsze implementacje

Najważniejsza jest klasa snake_algorithm

Klasa ta odpowiedzialna jest za algorytm gry. Zawiera ona dane dotyczące:

- Aktualnego położenia węża, owoców, przeszkód
- Kierunku poruszania się węża
- Rozmiaru planszy
- Wartości poszczególnych owoców
- Ilości przeszkód zostawianych przez węża (o ile ma je zostawiać)
- Możliwości przechodzenia przez ściany.

Klasa ta odpowiada również za poruszanie się węża. Zawiera ona tablicę board, która zawiera wszystkie elementy wyświetlane na planszy.

Konstruktor.

Parametry

X	ilość kolumn
Y	ilość wierszy
collision_Walls	określa czy wąż nie może przechodzić przez ściany
snake_Eggs	określa czy wąż ma zostawiać przeszkody
Fruit_1_Value	wartość owocu nr 1
Fruit_2_Value	wartość owocu nr 2
Fruit_3_Value	wartość owocu nr 3
eaten_Fruits_To_Grow	ilość owoców po których wąż zostawi przeszkodę

Definicja w linii 3 pliku **snake_algorithm.cpp**.

Ważniejsze składowe klasy snake_algorithm.

◆ move()

void snake_algorithm::move ( )

Metoda odpowiedzialna za poruszanie się węża.

Wywołanie tej metody powoduje przemieszczenie się węża o jeden kwadrat. Metoda ta sprawdza w jakim kierunku ma przemieścić się wąż, następnie dokonuje odpowiedniej zmiany współrzędnych pierwszego elementu ciała węża (głowy), po czym dodaje te współrzędne na początek tablicy **snake_coords** oraz usuwa ostatni element tej tablicy (wykonuje operacje **push** oraz **pop**). Jeśli ustawiono tryb gry w którym wąż może przechodzić przez ściany, to w momencie w którym miałoby to nastąpić współrzędne głowy zostaną tak zamienione, aby wąż znalazł się po przeciwnej stronie planszy.

Metoda ta porównuje również współrzędne głowy węża z elementem tablicy **board** o tych samych współrzędnych i zależnie od tego co znajduje się w tym elemencie:

- jeśli jest to owoc - zwiększy wynik o wartość owocu, wyśle sygnał **updateScore(int)**, nie usunie ostatniego elementu tablicy **snake_coords** (dzięki czemu wąż będzie większy o jeden element) oraz wywoła metodę **updateBoard()** (która doda współrzędne węża oraz przeszkód do **board**, tym samym nadpisując współrzędne owocu głową węża)
- jeśli jest to ściana - wartość **gameOver** zostanie ustawiona na **true**. Jeśli ustawiono możliwość przechodzenia przez ściany, to współrzędne głowy nigdy nie znajdą się na ścianie.
- jeśli to przeszkoda bądź ciało węża - wartość **gameOver** zostanie ustawiona na **true**.

Definicja w linii **76** pliku **snake_algorithm.cpp**.

◆ genFruit()

void snake_algorithm::genFruit ( ) private

Metoda generująca współrzędne owoców.

Metoda ta wywoływana jest w metodzie **move()** w momencie w którym wąż zjada owoc. Metoda ta zawsze generuje owoc nr 1, oraz losowo owoce nr 2 i 3. Metoda ta podczas generowania sprawdza czy współrzędne owocu są odpowiednie (tzn. kiedy odpowiadają one pustemu polu na tablicy **board**).

Definicja w linii **166** pliku **snake_algorithm.cpp**.

◆ genEgg()

void snake_algorithm::genEgg ( ) private

Metoda generująca przeszkody.

Metoda to wywoływana jest w metodzie **move()** w momencie kiedy wartość licznika **eaten_Fruits** osiągneła wymagany poziom. Współrzędne przeszkody równają się pierwszemu elementowi tablicy **snake_coords przed dodaniem elementu do tej tablicy** (czyli przed "przemieszczeniem się węża"), czyli po zakończu metody **move()** współrzędne odpowiadają drugiemu elementowi, co nie powoduję przegrania gry.

Definicja w linii **212** pliku **snake_algorithm.cpp**.

◆ updateBoard()

void snake_algorithm::updateBoard ( )

Aktualizacja tablicy **board**

Metoda ta dodaje do tablicy **board** elementy węża oraz przeszkody, których współrzędne przetrzymują odpowiednio **snake_coords** oraz **egg_coords**. Metoda ta pierw dodaje współrzędne przeszkody, potem dopiero współrzędne węża, tak aby móc je nadpisać, dzięki czemu podczas wyświetlania nie wystąpi sytuacja, w której jakiś element węża będzie przeszkodą.

Definicja w linii **65** pliku **snake_algorithm.cpp**.

◆ board

QVector< QVector<**content**> > snake_algorithm::board

Tablica przechowująca zawartość planszy gry.

Tablica ta w rzeczywistości jest wektorem wektorów typu wyliczeniowego **content**, który zawiera wszystkie elementy mogące znaleźć się na planszy gry. Dostęp do tablicy wygląda następująco: board[**indeks_wiersza**][**indeks_kolumny**].

Definicja w linii **106** pliku **snake_algorithm.h**.

Podsumowanie

Tworząc projekt zrealizowaliśmy prawie wszystkie początkowe założenia, wiele aspektów jest na otwartej drodze do rozbudowania.

Na koniec krótka lista naszych pomysłów na możliwe rozbudowanie lub poprawienie projektu:

- Dodanie opcji aktywacji dźwięku
- Zapisywanie stanu gry do pliku
- Dodanie animowanych motywów
- Przeniesienie gry na inne platformy
- Stworzenie trybu dla dwóch graczy
- Zastąpienie trzech okien jednym