**Kod źródłowy i potrzebne elementy**

Chińczyk został stworzony przy użyciu biblioteki graficznej w wersji 2.5.1 oraz bibliotek windows.h, iostream, string. Korzysta z plików graficznych w formacie .png (pionki w czterech kolorach, pola w pięciu kolorach i tło) oraz plików audio w formacie .wav (trzy melodie rozgrywki, melodia w menu, dźwięki zbicia i przesunięcia, oraz klaskanie dla zwycięzcy), które muszą znajdować się w ścieżce pliku .exe, aby zostały wczytane, podobnie jak plik czcionki „Kwadryga.ttf”, zawierający odpowiedni font. Przy kompilacji, należy pamietać dodatkowo folderze SFML-2.5.1, który musi się znajdować w ścieżce całego projektu. Kod źródłowy znajduje się w pliku Projekt C2.cbp. Program korzysta ze zmiennych globalnych ze względu na przejrzystość kodu. Nazwy funkcji oraz zmiennych nazwane są w języku Polskim.

**Zasada działania**

Program uruchamia się w trybie pierwszym, który wyświetla komunikat powitalny z nazwiskami autorów. Użytkownik naciska mysz, a program przechodzi do drugiego trybu, którym jest wybór ilości graczy (0-4). Nie ma możliwości wyboru koloru, gracze ludzcy przydzielani są do kolorów zgodnie z ruchem wskazówek zegara zaczynając od lewego górnego rogu (gracz czerwony). Trzeci etap to wybór szybkości wyświetlania komunikatów (0-3). Tryb 4 uruchamia właściwą rozgrywkę. Będzie trwał dopóki, któryś z gracz nie wygra partii, użytkownik wyłączy okno lub naciśnie przycisk Esc, który uruchamia tryb 1, bez względu na to w którym trybie aktualnie program się znajduje. Rozgyrwka opiera się kolejno na: zmianie gracza, jesli żaden z jego pionków nie może się ruszyć, sprawdzenie możliwości ruchu pionków nowego gracza, wybór pionka do ruszenia przez komputer lub użytkownika. Jeśli tura należy do gracza ludzkiego, program czeka na kliknięcie przycisku myszki, imitujące rzut kostki oraz oddanie tury po zakończeniu własnej. Przeciwnik komputerowy nie uruchamia takich wyzwalaczy. Program ma zaimplementowane również oszustwa przeznaczone wyłącznie dla programisty, gdyż używanie ich nie wyklucza błędów gry, ale przydatne są do testowania różnych sytuacji na planszy.

**Zmienne i funkcje**

**struct Pole** - Przechowuje informacje o polu na planszy gry:  
 > char znak - określa rodzaj pola (np. baza gracza, ścieżka, puste pole),  
 > int kolor, int numer - określają jakiego koloru i który numer ma pionek stojący na danym pole (jeśli pole jest zajęte). Przyjmują wartości od 1 do 4 (dla kolorów: 1 - czerwony, 2 - niebieski, 3 - żółty, 4 - zielony),  
 > sf::Sprite grafika - pozwala wyświetlić grafikę pola.  
**struct Pionek** - Przechowuje informacje o pionku:  
 > int miejsce - liczba określająca, w którym miejscu swojej trasy jest pionek (od 0 do 44)  
 > sf::Sprite grafika - p pozwala wyświetlić grafikę pionka.

**Pole pole[11][11]** - tablica będąca planszą, której każda komórka jest strukturą Pole. Poprzez tę tablicę program będzie zmieniał wartości na poszczególnych polach. Wielkość planszy to 11 x 11, ale nie wszystkie jej elementy uczestniczą w rozgrywce. Jeśli pole nie jest bazą, ścieżką lub metą, nie zostaje mu przypisana żadna grafika i pole jest niewidoczne.  
**Pionek pionek[4][4]** - tablica będąca zbiorem wszystkich pionków na planszy. Pierwsza współrzędna określa gracza i przyjmuje wartości od 0 do 3 (dlatego należy pamiętać, że przy jej wywoływaniu na podstawie zmiennej „gracz”, trzeba zmniejszyć ją o 1, gdyż ta przyjmuje wartości od 1 do 4). Pionki każdego gracza numerowane są od 0 do 3 (dlatego podobnie jak poprzednio, zmienną „wybrany” trzeba zmniejszać o 1 przy wyoływaniu, gdyż ta przyjmuje wartości od 1 do 4).  
**Pole \* trasa[4][4][45]** - tablica, która przechowuje trasy dla czterech pionków wszystkich czterech graczy (czyli łącznie 16 tras), zrobionych ze wskaźników na Pola w tablicy pole. Pierwsza i druga współrzędna podobnie jak w poprzedniej tablicy mówią o graczu i jego konkretnym pionku. Ostatnia przyjmująca wartości od 0 do 44 mówi o które miejsce w ścieżce konkretnie chodzi. Przez to program może dostać się na konkretne pole np. żeby sprawdzić czy konkretny pionek może się na nie przemieścić, znając aktualną pozycję tego pionka (zmienna „miejsce”).  
**int tak\_nie[4]** - tablica, do której zapisywane są możliwe ruchy pionka. Jeśli pionek może się ruszyć zapisywana jest jedynka, jeśli nie, zapisywane jest zero. Przykładowo: jeśli dla aktualnego gracza możliwość ruchu ma tylko pionek drugi, tablica będzie wyglądała następująco: {0,1,0,0}.

**int gracz** - zmienna określająca aktualnego gracza.  
**int kosc** - zmienna określająca wyrzuconą liczbę oczek.  
**int wybrany** - zmienna określająca pionek wybrany przez gracza.  
**int ilosc\_graczy** - zmienna określająca ilość graczy ludzkich, przyjmuje wartości od 0 do 4, a wiec możliwe jest uruchomienie partii, w której gra sam komputer (przydaje się to przy obserwacji działania gry).  
**int tryb** - zmienna określająca obszar, który ma być wykonywany w pętli głównej.  
**int skok\_czasu** - zmienna określająca długość przerwy między kolejnymi wyświetleniami komunikatów.  
**int wyzwalacz\_tura** - zatrzymuje program po zakończeniu tury gracza, by ten mógł ją oddać klikając przycisk myszy.  
**int wyzwalacz\_kostka** - zatrzymuje program przed rozpoczęciem tury gracza, by ten mógł „rzucić kostką” klikając przycisk myszy.  
**string tekst** - przechowuje komunikat do wyświetlenia.  
  
**sf::Texture tlo** - przechowuje grafikę tła.  
**sf::Sprite grafika\_tlo** - Pozwala wyświetlić grafikę tła.  
**sf::Texture pole0** - przechowuje grafikę białego pola.  
**sf::Texture pole1** - przechowuje grafikę czerwonego pola.  
**sf::Texture pole2** - przechowuje grafikę niebieskiego pola.  
**sf::Texture pole3** - przechowuje grafikę żółtego pola.  
**sf::Texture pole4** - przechowuje grafikę zielonego pola.  
**sf::Texture pionek1** - przechowuje grafikę czerwonego pionka.  
**sf::Texture pionek2** - przechowuje grafikę niebieskiego pionka.  
**sf::Texture pionek3** - przechowuje grafikę żółtego pionka.  
**sf::Texture pionek4** - przechowuje grafikę zielonego pionka.  
**sf::RenderWindow oknoAplikacji** - okno aplikacji  
**sf::Event event** - pozwala na obsługę wydarzeń takich jak użycie myszki lub klawiatury.  
**sf::Font MojFont** - przechowuje wczytany font „Kwadryga.ttf”  
**sf::Text Tekst** - pozwala na wyświetelnie „string tekst”.  
**sf::Music zbicie** - uruchamia dźwięk przy zbiciu obcego pionka.  
**sf::Music skok** - uruchamia dźwięk przy przesunięciu przyjaznego pionka.  
**sf::Music menu** - uruchamia melodią właściwą dla menu głównego.  
**sf::Music rozgrywka** - uruchamia jedną z trzech melodii właściwych dla rozgrywki.  
**sf::Music klaskanie** - uruchamia dźwięk klaskania przy zwycięstwie dowolnego gracza.

**void inicjacja()** - inicjuje wartości danych, które nie będą zmieniały się podczas rozgrywki. Wywołuje się tylko raz, na początku programu. Ustawia kolejno: punkt startowy dla mechanizmu generującego liczby pseudolosowe jako time; limit klatek na sekundę okienko na 120; tymczasową mapę znaków wielkości 11x11, która zostanie zapisana w zmiennych „znak” w komórkach tablicy „pole”, pozwalającą na łatwe modyfikowanie mapy przez programistę; ładuje grafikę pól, ustawia ich pozycję i skalę w oknie i zapisuje w tablicy „pole”; ładuje grafikę pionków, ustawia ich skalę i zapisuje ją do tablicy „pionek”; ładuje font „kwadryga.ttf”, ustawia jego kolor i pozycję; ładuje grafikę tła; ładuje wszelkie elementy audio, a dla elementu menu włącza zapętlenie.  
**void reset()** - wywoływana jest po każdej zakończonej rozgrywce oraz przy uruchomieniu programu ustawiając początkowe wartości: pionki umieszczane są w bazie, informacje o ich położone przywracane są do początkowych. Gracz ustawiany jest na czwartego, by zaraz po uruchomieniu partii zmienić się na pierwszego, ponieważ tablica „tak\_nie” czyszczona jest zerami.  
**int main()** - Zawiera główną pętlę programu. Przed nią uruchamia inicjację i reset. Pętla bez względu na zmienną „tryb” (ktorą określa wykonywany obszar pętli głównej) wywołuje funkcje „wylaczenie” i „wyswietlenie”. W trybie pierwszym zatrzymywane są dźwięki mogące być pozostałością po poprzedniej rozgrywce i uruchamiana jest melodia menu. Wyświetla się plansza witająca, a użytkownik przechodzi do kolejnego trybu klikając LPM. W trybie drugim użytkownik informowany komunikatem wybiera ilość graczy ludzkich (0-4) przy użyciu klawiatury i przechodzi do trybu trzeciego, a w nim ponownie informowany komunikatem wybiera przerwy między wyświetleniami komunikatów (0 - 0 milisekund, 1 - 100 milisekund, 2 - 500 milisekund, 3 - 1000 milisekund), a przy wyborze zatrzymywana jest melodia menu. Tryb czwarty jest już właściwą grą. Puszcza jedną losowo wybraną melodię spośród trzech przeznaczonych do rozgrywki, robiąc to ponownie kiedy ta się skończy. Pierwszy if sprawdza czy aktualny gracz ma możliwość ruchu i w zależności od tego, program wykonuje szereg różnych operacji. Jeśli można wykonać ruch: sprawdza się czy zostały użyte oszustwa; wywołuje się funkcję „wybor” (gracz<=ilość\_graczy) lub „wybor\_komputer” ” (gracz>ilość\_graczy) w zależności czy tura należy do gracza czy do komputera; sprawdza czy któryś gracz wygrał. Jeśli nie można wykonać ruchu: jeśli tura należała do gracza, program czeka na jego kliknięcie pozwalające kontynuować; jeśli w turze nie wypadła liczba oczek 6 lub gracz nie zbił innego gracz, kolej przechodzi na następnego gracza; wyświetlany jest komunikat o aktualnym graczu; jeśli nowa tura należy do gracza ludzkiego program czeka na jego kliknięcie imitujące rzut kostką; losowana jest liczba oczek na kostce; sprawdzane są możliwości ruchu pionków aktualnego gracza. W pętli głównej znajduje się jeszcze tryb zerowy, który uruchamia się gdy któryś z graczy wygra pojedynek. Wtedy też wywołuje się funkcję reset a użytkownik musi nacisnąć przycisk by przejść do trybu pierwszego.  
**void wyswietl()** - funkcja wywoływana przy każdym przejściu głównej pętli oraz przy wyświetlaniu kolejnych komunikatów. Ładuje tekst do wyświetlenia. Nanosi na okno aplikacji kolejno: tło, planszę, pionki, tekst i wyświetla okno aplikacji.  
**void sprawdzenie()** - wywoływane przy każdej zmianie gracz, wykonaniu przez niego ruchu pozwalającego na kolejnych ruch lub przy użyciu oszustwa. Wyświetla liczbę oczek kostki komunikatem i zaczyna sprawdzenie. Inicjuje tablicę „tak\_nie” jedynkami i sprawdza wszystkie pionki gracza. Jeśli sprawdzany pionek nie ma możliwości, zapisywane jest mu z powrotem zero. Sytuacje kiedy pionek nie może się ruszyć: pionek jest w bazie, a ilość oczek na kostce jest inna niż 6; pionek jest w bazie, a liczba oczek to 6, ale na wyjściu już stoi przyjazny pionek; pionek jest już na ostatnim miejscu mety; pionek przeskoczyłby metę; pionek wszedłby na przyjazny pionek. Na koniec wyświetla odpowiedni komunikat, czy gracz może wybrać pionek, czy nie ma takiej możliwości.  
**void wybor()** - pozwala graczowi ludzkiemy wybrać prawidłowy pionek. Funkcja wywołuje się tylko jeśli chociaż jeden pionek ma możliwość ruchu. Funkcja zbierając informację o aktualnym położeniu myszki sprawdza czy pole nad którym znajduje się kursor zawiera pionek w golorze aktualnego gracza. Jeśli tak, sprawdz czy ten pionek ma możliwości ruchu przy pomocy tablicy „tak\_nie”. Po prawidłowym wybraniu pionka, zostaje wywołana funkcja ruch(), a tablica „tak\_nie” zostaje wyzerowana.  
**void wybor\_komputer()** - pozwala komputerowi wybrać prawidłowy pionek. Funkcja wywołuje się tylko jeśli chociaż jeden pionek ma możliwość ruchu, a tura należy do gracza komputerowego. Losowana jest liczba od 1 do 4 dopóki nie trafi pionka z tym numerem, który może się ruszyć. Po prawidłowym wybraniu pionka, zostaje wywołana funkcja ruch(), a tablica „tak\_nie” zostaje wyzerowana.  
**void ruch()** - wywołana tylko przed dwie poprzednie funkcje. Uruchamia dźwięk skoku i zeruje aktualne miejsce na planszy. Jeśli pionek stoi w bazie, zmienna „miejsce” zmienia wartość z 0 na 1. Jeśli jednak pionek jest już na planszy, przesuwa się o tyle pól ile wynosi liczba oczek kostki. W sytuacji napotkania wrogiego pionka, ten jest przesuwany na swoją zerową pozycję, razem z informacją o jego położeniu na planszy. Na koniec zapisywane są informacje na planszy o pionku aktualnego gracza, który zmienił pozycję.  
**void wygrana()** - sprawdza czy któryś z graczy wygrał. Jeśli któryś z nich ma wszystkie pionki ustawione na linii mety, wyświetlany jest komunikat, włączany jest dźwięk oklasków, a tryb głównej pętli zmienia się na zerowy.  
**void oszustwo()** - jeśli użytkownik naciśnie strzałkę w lewo, liczba oczek zmniejszy się o jeden, jeśli w prawo, zwiększy się o jeden. Strzałka w górę oznacza liczbę 40 oczek na kostce. Po każdym uruchomieniu oszustwa wywołuje się funkcja sprawdzenie().  
**void wylaczenie()** - funkcja pozwala na wyłączenie całego okna aplikacji, przy użyciu przycisku zamknięcia w prawym górnym rogu lub przejście do tytułowej planszy (czyli trybu pierwszego), bo naciśnięciu klawisza „Esc” przez użytkownika.