# **Project “Checkers” Dawid Podgórski**

Przedmiotem mojego projektu jest gra warcaby. Moja gra reprezentuje rozgrywkę w warcaby klasyczne, plansza składa się z 64 pól 8x8 oraz 12 pionków białych oraz 12 czarnych. Wszelkie zasady gry zostały zaimplementowane zgodnie ze standardem tej odmiany warcabów.

Program obejmuje menu główne w którym można wybrać typ rozgrywki, mamy do wyboru gracz vs gracz oraz gracz vs komputer. Przy pierwszym wyborze algorytm wygeneruje od razu planszę do gry natomiast jeżeli dokonamy wyboru gry z botem zostanie załadowane menu wyboru koloru. Z tego poziomu można także wrócić do menu głównego.

Interfejs graficzny gry został zaimplementowany z wykorzystaniem biblioteki „SFML”. Projekt jest podzielony na trzy klasy: „Board”, „Pawn”, „Game”.

Klasa Pawn jest klasą bazową, najbardziej podstawową z klas. Zawiera w sobie struktury tj. Field oraz Moves i Capture. Są one umieszone tutaj ponieważ każdy pionek musi zawierać takie właściwości. Field jest odpowiedzialny za przydzielenie do pionka lokalizacji, Moves zawiera w sobie ruchy jakie może podjąć pionek. Zawiera w sobie lokalizację ruchu oraz wskaźnik na pionek, jest ona rozwinięta o prawdopodobieństwo, które służy do określenia opłacalności ruchu przy grze z botem. Capture zawiera w sobie wskaźnik na posunięcie po biciu oraz wskaźnik na pole na którym znajduje się pionek do zbicia, jest to niezwykle pomocne w dalszej implementacji. Do pól klasy należy string odpowiadający za nazwę pionka, wskaźnik na lokalizację, wektor możliwych ruchów, jest to wektor typu Moves. Kolor piona, rodzaj pionka, jego sprite i status na planszy a także wektor możliwych bić. W konstruktorze klasy wczytane są tekstury pionków.

Ta klasa dziedziczy publicznie po klasie Pionek. Polami klasy Plansza są podstawowe tekstury, ich sprity, wysokość, szerokość oraz licznik odpowiedzialny za zliczenie posunięć damami bez zmiany stanu liczby pionków na planszy. W konstruktorze tej klasy załadowują się tekstury z plików. Klasa jest też wyposażona w metodę inicjalizującą planszę. Ta metoda odpowiada za ustawienie obiektów na planszy oraz ustawienie wartości pól poszczególnych obiektów składowych jest ona rozwinięta w klasie Game.

Klasa Game dziedziczy publicznie po planszy, jest najbardziej złożoną klasą. Zawiera w sobie strukturę Player która odpowiada za opis obecnego gracza. Zawiera w sobie kolor, nazwę oraz rodzaj gracza, który jest to najbardziej istotny podczas rozgrywki gracz vs komputer. Do jej pól należą tekstury menu głównego, menu wyboru kolory oraz ich sprity. Zmienne odpowiadające za ostatnią pozycję myszy służące do precyzyjnego przemieszczania pionka względem myszki. Pole gracza pierwszego oraz gracza drugiego określające ich specyfikację. Wskaźnik na pierwsze pole planszy oraz ostatnie. Wektor możliwych posunięć, wskaźnik na aktywnego pionka, zmienne określające stan rozgrywki oraz wskaźnik na pionka który ostatni bił, szczególnie przydatny dla implementacji bić wielokrotnych. Ostatnim polem jest zmienna typu Player która określa, który gracz obecnie wykonuje ruch.

Jeśli chodzi o metody tej klasy, to isMouseOverPawn odpowiada za sprawdzenie czy myszka jest nad pionkiem, który obecnie może wykonać ruch na podstawie wektora określającego możliwe ruchy gracza. Bicie jest zrealizowane w trzech metodach. Pierwsza z nich ładuję wszystkie pionki gracza, który obecnie podejmuje ruch do drugiej metody. Na tym etapie sprawdzany jest rodzaj pionka, aby oddelegować go do odpowiedniej metody. Pionki zwykłe trafiają do metody która sprawdza czy mają bicia na dwóch polach do przodu lub do tyłu damy do metody sprawdzająca bicia względem jego specyfiki. Trzeci etap jest wykonywany na późniejszym etapie gry w którym to gracz upuszcza pionka. Metoda odpowiadająca za wygenerowanie ruchu sprawdza możliwość ruchu dla danych pionków. Metoda pawnIsPossibleSpace sprawdza czy pionek został upuszczony na odpowiednie dostępne pole zawarte w wektorach posunięć lub bić. Metoda changePlayer odpowiada za zmianę gracza. Metoda turnIntoQueen odpowiada za zmianę pionka w damę jeśli spełni określone warunki zawarte w metodzie. moveQueen oraz captureQueen to metoda analogiczne do tych związanych z pionkiem zwykłym ale zaimplementowane z myślą o damce. isItOver sprawdza czy sprawdza czy zostały spełnione warunki do zakończenia gry oraz określa jak zostanie zakończona. Move jest metodą obsługującą zdarzenia. Jest także metoda odpowiadająca za zmianę ostatniej pozycji myszki. Metody display są odpowiedzialne za wyświetlanie zawartości planszy oraz jej samej oraz Menu. Metody choice odpowiedzialne za obsługę zdarzeń na pewnym etapie gry. Metoda inicjalizująca planszę w klasie Game rozwija metodę z klasy Board o dodatkowe wartości. W konstruktorze wczytane są tekstury oraz ustawione ich wielkości.