Użyte w tym dokumencie słowa MUSI, POWINIEN, MOŻE, NIE MOŻE należy rozumieć zgodnie z RFC 2119

1 Logika gry

Główna logika gry będzie się znajdowała w klasie `Game`, która będzie przechowywała stan gry wraz z kompletną historią ruchów (jako instancję klasy `GameState`) oraz informacje pomocnicze. Będzie ona trzymać dwa wskaźniki na interfejsy graczy (klasy `Player`). Każdy gracz MUSI obsługiwać funkcje `GenMove()` oraz `GameUpdated()`, które oznaczają odpowiednio żądanie wygenerowania ruchu oraz informację o tym, że plansza uległa zmianie. Każda z tych funkcji MUSI w miarę szybko zwrócić, być może uruchamiając sobie dodatkowy wątek do dłuższych obliczeń. Gracz informuje o swoim ruchu wołając metodę `Move(int direction)` na rzecz klasy `Game` (wskaźnik na bieżącą grę jest trzymany w klasie `Player`).

Gra dysponuje własną pętlą gry, uruchamianą za pomocą metody `Start()` i przerywanej metodą `Stop()`. Działa ona w osobnym wątku i jest zatrzymywana na semaforze na czas oczekiwania na ruch gracza. Po podniesieniu semaforu przez metodę `Move`, wątek ten znowu jest zatrzymywany na semaforze, tym razem w oczekiwaniu na reakcję nadzorcy (instancji klasy `UI`), który MUSI albo zadecydować o przerwaniu pętli gry lub podniesieniu semaforu metodą `Continue` na rzecz gry.

Gracz ma prawo pytać się gry o dostępne mu ruchy (funkcja `ValidMoves`) oraz pytać się, czy konkretny ruch jest dozwolony (funkcja `IsMoveValid`).

Przy uruchamianiu nowej gry obiekty klasy `Player` są niszczone, więc MUSZĄ one w destruktorze zwolnić wszystkie swoje zasoby, w szczególności utworzone wątki.

Gra można znajdować się w jednym ze stanów: `EDIT`, `HINT`, `ACTIVE`, `PLAYER1_WON`, `PLAYER2_WON`.

2 Przechowywanie segmentów

Segment trzymany jest w dedykowanej strukturze danych, zawierającej informacje o początkowej pozycji piłki, kierunku ruchu i graczu, który go postawił. Wszystkie segmenty są przechowywane w strukurze `Index`. Segmenty niepostawione przez żadnego gracza mają ustawionego gracza na wartość `force_majeure`.

3 Interfejs graficzny

Główną kontrolką jest klasa `FootballCtrl`, która przyjmuje żądania od użytkownika oraz obsługuje animację piłki. Może ona także utworzyć obiekt klasy dziedziczącej z klasy `Player`, który będzie przekazywał wszystkie żądania gry do kontrolki.

Oprócz tego istnieje klasa `GameProperties`, która reprezentuje okno pytania się użytkownika o parametry nowej gry: wymiary planszy oraz typy graczy (człowiek/komputer).

Główne okno jest reprezentowane przez klasę `Football`, która przechowuje obiekty klas `Game`, `UI`, `FootballCtrl` i w zależności od wskazanych przez użytkownika opera-

cji (jak utworzenie nowej gry) ustawia odpowiednie wartości tych obiektów, dbając o uruchamianie/wstrzymywanie wątku gry.

Edycja planszy następuje na tej samej kontrolce `FootballCtrl`, na której odbywa się gra.

4 Format danych

Gra jest zapisywana do pliku w formie pliku XML, tworzonego standardowymi narzędziami serializacji dostępnymi w U++. Główny węzeł ma nazwę `game`, jego synami są węzły `player1` i `player2` mające atrybut `value` równy `0` jeśli gracz jest człowiekiem i `1` w przeciwnym wypadku.

Jego synem jest także węzeł `state` opisujący stan gry. Zawiera on węzły `height` i `width` mające jako atrybut `value` odpowiednio wysokość i szerokość planszy. Oprócz tego zawiera węzeł `ballPosition`, który w atrybutach `x`, `y` zawiera pozycję piłki. Węzeł `currentPlayer` ma w atrybucie `value` wartość `player1` lub `player2`, z oczywistym znaczeniem.

Synem węzła `state` jest także węzeł `segments`, który zawiera opis wszystkich istniejących na planszy segmentów, wraz z jej brzegiem, w kolejności ich dodawania do planszy.

Każdy segment jest reprezentowany przez węzeł `key`, trzymający trzech synów:

- `position` (które w atrybutach `x` oraz `y` trzyma pozycję piłki w momencie początku ruchu),
- `direction` (który w atrybucie `value` trzyma kierunek ruchu: liczbę od 0 do 8) oraz
- `player`, które zawiera identyfikator gracza, który postawił ten segment: `player1` lub `player2`. Jeśli segment został dodany edytorem, znajduje się tam `force_majeure`.

5 Podział na moduły

- Logika gry znajduje się w plikach `Game.{h,cpp}`
- kontrolka `FootballCtrl` znajduje w plikach `FootballCtrl.{h,cpp}`
- okna znajdują się w plikach `Football.h` oraz `main.cpp`,
- sztuczna inteligencja w pliach `AI.{h,cpp}`