2015

Warcaby

Dokumentacja Projektu

Wprowadzenie do Dokumentacji, Opis Klas, Diagramy, Testy jednostkowe

Jarosław Biskupski, Krzysztof Matysik, Jakub Buziałkowski
Michał Miszewski, Kamil Pałagan
Warcaby
2015-05-28

Zawartość

1 Wstęp - Wprowadzenie do dokumentacji 5

1.1 Definicje i skróty 5

1.1.1 Rozgrywka - jest to główna funkcjonalność aplikacji, która odpowiada za obsługę zdarzeń związanych z przebiegiem tury gry w warcaby pomiędzy dwoma użytkownikami 5

1.1.2 User-friendly - przyjazny użytkownikowi 5

1.1.3 GUI - graficzny interfejs użytkownik 5

1.2 Charakterystyka użytkowników (Aktorów) 5

1.2.1 Gracz - Końcowy odbiorca projektu, typ użytkownika grający w trybie samouczka lub pierwszy gracz w trybie - turowym 5

1.2.2 Gracz II - drugi gracz uczestniczący w trybie wieloosobowym - turowym 5

1.3 Ogólne możliwości aplikacji 5

1.3.1 Przeprowadzenie rozgrywki dwuosobowej na jednym urządzeniu (system turowy) 5

1.3.2 Odtwarzanie prostych dźwięków związanych z ruchami pionków na szachownicy 5

1.3.3 Odtwarzanie muzyki w tle 5

1.3.4 Samouczek przedstawiający zasady gry w warcaby 5

1.3.5 Możliwość wyboru stylu graficznego szachownicy 5

1.3.6 Możliwość wyboru stylu graficznego pionków 5

1.3.7 Zmiana perspektywy widoku 5

1.4 Ogólne ograniczenia. 5

1.4.1 Użytkownik musi posiadać urządzenie z systemem Android w wersji co najmniej 4.0.3 5

1.4.2 Urządzenie musi być wyposażone w interfejs bluetooth w wersji 2.0 lub wyższej 5

1.4.3 Urządzenie musi posiadać minimum 512 MB pamięci operacyjnej oraz procesor taktowany z częstotliwością 800 MHz lub wyższej 5

1.4.4 Aplikacja potrzebuje około 100MB pamięci wewnętrznej 5

2 Dokumentacja struktur danych 6

2.1 Dokumentacja klasy Board 6

2.1.1 Metody publiczne 6

2.1.2 Pola danych 6

2.1.3 Opis szczegółowy 6

2.1.4 Dokumentacja funkcji składowych 6

2.2 Dokumentacja klasy ButtonsControl 8

2.2.1 Metody publiczne 8

2.2.2 Pola danych 8

2.2.3 Opis szczegółowy 8

2.2.4 Dokumentacja funkcji składowych 8

2.3 Dokumentacja klasy CameraPerspective 9

2.3.1 Metody publiczne 9

2.3.2 Pola danych 9

2.3.3 Opis szczegółowy 9

2.3.4 Dokumentacja funkcji składowych 9

2.4 Dokumentacja klasy Cursor 9

2.4.1 Dodatkowe Dziedziczone Składowe 10

2.5 Dokumentacja klasy Field 10
2.5.1 Metody publiczne 10
2.5.2 Pola danych 10
2.5.3 Opis szczegółowy 10
2.5.4 Dokumentacja pól 10
2.6 Dokumentacja struktury FieldCords 11
2.6.1 Pola danych 11
2.7 Dokumentacja klasy Gameplay 11
2.7.1 Metody publiczne 11
2.7.2 Pola danych 11
2.7.3 Opis szczegółowy 11
2.7.4 Dokumentacja funkcji składowych 12
2.7.5 Dokumentacja pól 13
2.8 Dokumentacja klasy HintWindow 13
2.8.1 Opis szczegółowy 13
2.9 Dokumentacja klasy Menu 13
2.9.1 Metody publiczne 13
2.9.2 Pola danych 14
2.9.3 Opis szczegółowy 14
2.9.4 Dokumentacja funkcji składowych 14
2.10 Dokumentacja klasy MenuBoard 15
2.10.1 Pola danych 15
2.10.2 Opis szczegółowy 15
2.10.3 Dokumentacja pól 15
2.11 Dokumentacja klasy Options 15
2.11.1 Metody publiczne 15
2.11.2 Pola danych 16
2.11.3 Opis szczegółowy 16
2.11.4 Dokumentacja pól 16
2.12 Dokumentacja klasy Pawn 16
2.12.1 Metody publiczne 16
2.12.2 Pola danych 17
2.12.3 Opis szczegółowy 17
2.12.4 Dokumentacja funkcji składowych 17
2.12.5 Dokumentacja pól 17
2.13 Dokumentacja klasy Player 18
2.13.1 Metody publiczne 18
2.13.2 Pola danych 18
2.13.3 Opis szczegółowy 18
2.13.4 Dokumentacja funkcji składowych 18
2.14 Dokumentacja klasy Tutorial 19
2.14.1 Metody publiczne 19

2.14.2 Opis szczegółowy 19
2.14.3 Dokumentacja funkcji składowych 20
3 Podsumowanie 21
3.1 Aplikacja działa stabilnie i poprawnie oraz spełnia wszystkie wymagania funkcjonalne oraz nie funkcjonalne - rozgrywka jest satysfakcjonująca 21
3.2 Apliakacja zostałą umieszczona w sklepie Google Play w celu jej dystrybucji 21
3.3 Aplikacja działa na systemach Android 4.0.3 i nowszych 21
3.4 Aplikacja została wykonana w środowisku Unity , wykorzystywany był język programowania C# .. 21
3.5 Testy zostały przeprowadzone pomyślnie, napotkane błędy usunięte, ... 21
3.6 Odpowiednie diagramy i tabele zostały dodane w załączniku w celu zwiększenia ich czytelności 21
4 Indeks 22
5 Załączniki 24
5.1 Diagram Klas 24
5.2 Schematy blokowe użyte podczas projektowania aplikacji: 26
5.3 Diagram Przepływu danych: 28

1 Wstęp - Wprowadzenie do dokumentacji

Dokemunt ten zawiera dokumentację do projektu "Warcaby", którego celem było stworzenie aplikacji - gry na system Android 4.0.3 i nowszy. Aplikacja spełnia wszystkie wymogi , które były zawarte w "System Requirements Secification for "Warcaby".

1.1 Definicje i skróty

1.1.1 Rozgrywka - jest to główna funkcjonalność aplikacji, która odpowiada za obsługę zdarzeń związanych z przebiegiem tury gry w warcaby pomiędzy dwoma użytkownikami

1.1.2 User-friendly - przyjazny użytkownikowi

1.1.3 GUI - graficzny interfejs użytkownik

1.2 Charakterystyka użytkowników (Aktorów)

1.2.1 Gracz - Końcowy odbiorca projektu, typ użytkownika grający w trybie samouczka lub pierwszy gracz w trybie - turowym

1.2.2 Gracz II - drugi gracz uczestniczący w trybie wieloosobowym - turowym

1.3 Ogólne możliwości aplikacji

1.3.1 Przeprowadzenie rozgrywki dwuosobowej na jednym urządzeniu (system turowy)

1.3.2 Odtwarzanie prostych dźwięków związanych z ruchami pionków na szachownicy

1.3.3 Odtwarzanie muzyki w tle

1.3.4 Samouczek przedstawiający zasady gry w warcaby

1.3.5 Możliwość wyboru stylu graficznego szachownicy

1.3.6 Możliwość wyboru stylu graficznego pionków

1.3.7 Zmiana perspektywy widoku

1.4 Ogólne ograniczenia.

1.4.1 Użytkownik musi posiadać urządzenie z systemem Android w wersji co najmniej 4.0.3

1.4.2 Urządzenie musi być wyposażone w interfejs bluetooth w wersji 2.0 lub wyższej

1.4.3 Urządzenie musi posiadać minimum 512 MB pamięci operacyjnej oraz procesor taktowany z częstotliwością 800 MHz lub wyższej

1.4.4 Aplikacja potrzebuje około 100MB pamięci wewnętrznej

2 Dokumentacja struktur danych

2.1 Dokumentacja klasy Board

Klasa Planszy
Dziedziczy MonoBehaviour.

2.1.1 Metody publiczne

List< GameObject > **getDarkFieldsList** ()
Zwraca listę ciemnych pól

FieldCords TranslateCords (string sID)
Funkcja zamienia ID pola ze stringa do postaci współrzędnych

void **setFieldState** (FieldState fStat, string idField)
Ustawia status pola

FieldState **getFieldState** (string idField)
zwraca status pola

Vector3 **getFieldCenterCoordinate** (string idField)
zwraca współrzędne środka pola

int **getPawnIDonField** (string idField)
string **getFieldIDinLine** (string startFieldID, string middleFieldID)
Zwraca id pola w lini prostej, jeśli pole jest zbyt blisko krawędzi planszy to zwraca napis brak_pola

bool **isItFront** (string sFieldID, string eFieldID, int pID)
Sprawdza czy pole sąsiadujące z pionkiem jest z przodu

List< string > **getSurroundingFields** (string idField)
Funkcja zwraca id-ki pól sąsiadujących z polem o podanym id jako argument

List< string > **getSurroundingFieldsWithEnemies** (string idField, int pID)
List< string > **getFrontFields** (string idField, int pID)
bool **searchForCaptures** (**Pawn** cPawn)
bool **areThereCaptures** (**Pawn** cPawn)

2.1.2 Pola danych

GameObject **Field_bright**
GameObject **Field_dark**
GameObject[][] **Fields**

2.1.3 Opis szczegółowy

Klasa Planszy

2.1.4 Dokumentacja funkcji składowych

2.1.4.1 bool Board.areThereCaptures (Pawn *cPawn*)

Odczytujemy id pól znajdujących się w sąsiedztwie pionka, na których stoi wróg
Sprawdzamy czy pola za pionkiem wroga w lini prostej są wolne
Najpierw szukamy id pola za pionkiem wroga
Trzeba uzyskać id pionka wroga do bicia
ID pionka trzeba przetłumaczyć na index

Sprawdzamy czy to pole jest puste

2.1.4.2 List<GameObject> Board.getDarkFieldsList ()

Zwraca listę ciemnych pól

2.1.4.3 Vector3 Board.getFieldCenterCoordinate (string *idField*)

zwraca współrzędne środka pola

Konwertujemy id pola
Zwracamy pozycję pola jako Vector3

2.1.4.4 string Board.getFieldIDinLine (string *startFieldID*, string *middleFieldID*)

Zwraca id pola w lini prostej, jeśli pole jest zbyt blisko krawędzi planszy to zwraca napis brak_pola

2.1.4.5 FieldState Board.getFieldState (string *idField*)

zwraca status pola

zwracamy status pola

2.1.4.6 int Board.getPawnIDonField (string *idField*)

Konwertujemy ID pola

2.1.4.7 List<string> Board.getSurroundingFields (string *idField*)

Funkcja zwraca id-ki pól sąsiadujących z polem o podanym id jako argument

2.1.4.8 List<string> Board.getSurroundingFieldsWithEnemies (string *idField*, int *pID*)

Sprawdzamy czy coś stoi na polu
Sprawdzamy czy to pionek wroga

2.1.4.9 bool Board.isItFront (string *sFieldID*, string *eFieldID*, int *pID*)

Sprawdza czy pole sąsiadujące z pionkiem jest z przodu

2.1.4.10 bool Board.searchForCaptures (Pawn *cPawn*)

Sprawdzamy czy pola za pionkiem wroga w lini prostej są wolne
Najpierw szukamy id pola za pionkiem wroga
Trzeba uzyskać id pionka wroga do bicia
ID pionka trzeba przetłumaczyć na index

Zabezpieczenie przed sytuacją kiedy nie ma możliwości bicia pionków, wtedy musi nastąpić zmiana tury

Sprawdzamy czy to pole jest puste

Jeśli tak to podświetlamy

Zapamiętujemy id pionka ktory wykonuje bicie

2.1.4.11 void Board.setFieldState (FieldState *fStat*, string *idField*)

Ustawia status pola

Konwertujemy ID pola

Ustawiamy stan dla pola podany w argumencie

2.1.4.12 FieldCords Board.TranslateCords (string *sID*)

Funkcja zamienia ID pola ze stringa do postaci współrzędnych

2.1.4.13 Dokumentacja dla tej klasy została wygenerowana z pliku:

gameplay/Board.cs

2.2 Dokumentacja klasy ButtonsControl

Klasa Buttons.Control

Dziedziczy **Menu**.

2.2.1 Metody publiczne

void **PlayClickSound** ()

Odtwarzanie dzwięku

2.2.2 Pola danych

AudioClip **click**

2.2.2.1.1 Parametry:

click	Dzwięku "clicku" n

2.2.3 Opis szczegółowy

Klasa Buttons.Control

2.2.4 Dokumentacja funkcji składowych

2.2.4.1 void ButtonsControl.PlayClickSound ()

Odtwarzanie dzwięku

2.2.4.2 Dokumentacja dla tej klasy została wygenerowana z pliku:

ButtonsControl.cs

2.3 Dokumentacja klasy CameraPerspective

Klasa Odpoweidzialna za ustawienia kamery
Dziedziczy MonoBehaviour.

2.3.1 Metody publiczne

void **CameraZoomIn** ()
void **CameraZoomOut** ()
void **setTopView** ()
void **setFreeView** ()

2.3.2 Pola danych

GameObject **buttontopview**
Camera **camera**
float **ZoomSpeed**

2.3.3 Opis szczegółowy

Klasa Odpoweidzialna za ustawienia kamery

2.3.4 Dokumentacja funkcji składowych

2.3.4.1 void CameraPerspective.setFreeView ()

Ustawienie elementów interfejsu
Przywrócenie kamery do poprzedniego widoku

2.3.4.2 void CameraPerspective.setTopView ()

Ustawienie elementów interfejsu
Zapamiętanie pozycji kamery
Ustawienie kamery nad planszą

2.3.4.3 Dokumentacja dla tej klasy została wygenerowana z pliku:

gameplay/CameraPerspective.cs

2.4 Dokumentacja klasy Cursor

Dziedziczy **Gameplay**.

2.4.1 Dodatkowe Dziedziczone Składowe

Dokumentacja dla tej klasy została wygenerowana z pliku:
gameplay/Cursor.cs

2.5 Dokumentacja klasy Field

Klasa Pola Gry
Dziedziczy MonoBehaviour.

2.5.1 Metody publiczne

void **setHighlighted** ()
void **unsetHighlighted** ()

2.5.2 Pola danych

string **idField**
Identyfikator pola
FieldState **fState**
Stan pola
int **pawnId**
Identyfikator pionka który stoi na polu

2.5.3 Opis szczegółowy

Klasa Pola Gry

2.5.4 Dokumentacja pól

2.5.4.1 FieldState Field.fState

Stan pola

2.5.4.2 string Field.idField

Identyfikator pola

2.5.4.3 int Field.pawnId

Identyfikator pionka który stoi na polu

2.5.4.4 Dokumentacja dla tej klasy została wygenerowana z pliku:

gameplay/Field.cs

2.6 Dokumentacja struktury FieldCords

2.6.1 Pola danych

int **Y**
int **X**

Dokumentacja dla tej struktury została wygenerowana z pliku:
gameplay/Board.cs

2.7 Dokumentacja klasy Gameplay

Klasa **Gameplay**
Dziedziczy MonoBehaviour.
Dziedziczona przez **Cursor**.

2.7.1 Metody publiczne

void **onTurnStart** (int pID)
Funkcja musi być wywoływana na początku każdUj tury aby sprawdzić czy są ruchy i bicia dla danego gracza

GameObject **getEnemyPlayer** ()
Zwraca gracza który posiada prawo ruchu

void **changeTurn** ()
Funkcja zmieniająca gracza

void **showResults** (string Winner)
void **backToMainMenu** ()

2.7.2 Pola danych

int **whoseTurnID**
Elementy funkcjonalne

bool **isThereCapture**
bool **isThereMove**
GameObject **panelGameEnd**
Elementy GUI

GameObject **panelGameStart**
Text **textWinner**
Image **wPawnS**
GameObject **Field_bright**
GameObject **Field_dark**
Mesh **boardMesh**
Material **boardMaterial**

2.7.3 Opis szczegółowy

Klasa **Gameplay**

2.7.4 Dokumentacja funkcji składowych

2.7.4.1 void Gameplay.backToMainMenu ()

Sprawdzamy platformę i zamykamy program

2.7.4.2 void Gameplay.changeTurn ()

Funkcja zmieniająca gracza

2.7.4.3 GameObject Gameplay.getEnemyPlayer ()

Zwraca gracza który posiada prawo ruchu

2.7.4.4 void Gameplay.onTurnStart (int *pID*)

Funkcja musi być wywoływana na początku kaźdUj tury aby sprawdzić czy są ruchy i bicia dla danego gracza

Ustawiamy wartość określającą czy jest jakieś bicie na fałsz
Ustawiamy wartość określającą czy jest jakiś ruch na fałsz
Pobieramy listę pionków gracza
Trzeba sprawdzić każdy pionek pod względem możliwości wykonania ruchu bądź bicia
Pobieramy komponent skryptu dla aktualnie sprawdzanego pionka
Sprawdzamy czy pionek nie został ubity
Trzeba pobrać id pola na którym jest pionek
Tworzymy listę która będzie przechowywać id-ki sąsiadujących pól
W tej pętli sprawdzane będą wszystkie pola sąsiadujące, czy na którymś jest pionek wroga
Jeśli pole jest nie puste
Jeśli pionek wroga
Sprawdzamy czy w lini prostej za tym polem jest wolne pole
Trzeba sprawdzić czy pole za jest wolne dlatego pobieramy status pola końcowego
sprawdzamy czy pole końcowe jest puste
Jeśli pole za pionkiem wroga jest puste to mamy bicie
Jeśli pole graniczne jest puste
Sprawdzamy czy pole graniczne jest z przodu
jeśli pole jest z przodu to ustawiamy MOVE_ALLOWED
a pionek moze wykonać ruch jeśli pionek ma bicie to nie może wykonać ruchu
Sprawdzamy czy są jakieś ruchy lub bicia
Jeśli nie ma możliwych ruchów lub bić to koniec gry i remis

2.7.4.5 void Gameplay.showResults (string *Winner*)

Wyświetlenie remisu
Wyświetlenie informacji o tym który gracz wygrał

2.7.5 Dokumentacja pól

2.7.5.1 GameObject Gameplay.panelGameEnd

Elementy GUI

2.7.5.2 int Gameplay.whoseTurnID

Elementy funkcjonalne

2.7.5.3 Dokumentacja dla tej klasy została wygenerowana z pliku:

gameplay/Gameplay.cs

2.8 Dokumentacja klasy HintWindow

Klasa **HintWindow**
Dziedziczy MonoBehaviour.

2.8.1 Opis szczegółowy

Klasa **HintWindow**

Dokumentacja dla tej klasy została wygenerowana z pliku:
Tutorial/HintWindow.cs

2.9 Dokumentacja klasy Menu

Klasa **Menu** Gównego
Dziedziczy MonoBehaviour.
Dziedziczona przez **ButtonsControl**.

2.9.1 Metody publiczne

void **ExitApplication** ()
Sprawdzenie platformy i zamknięcie aplikacji
void **CreateGameOnOneDevice** ()
Przejście do nowej sceny - utworzenie rozgrywki na jednym urządzeniu
void **CreateGame** ()
Przejście do nowej sceny - utworzenie rozgrywki
void **JoinGame** ()
Przejście do nowej sceny - dołączenie do rozgrywkui
void **StartTutorial** ()
Przejście do nowej sceny - samouczka
void **changeMusicVolume** ()

void **changeSoundsVolume** ()
void **changeGraphicQuality** ()
void **changeShadows** ()
void **Info** ()
Pokazujemy panel z informacjami
void **InfoExit** ()
Ukrywamy panel z informacjami

2.9.2 Pola danych

GameObject **mainCanva**
GameObject **options**
InputField **inputGameName**
ToggleGroup **toggleColorGroup**

2.9.3 Opis szczegółowy

Klasa **Menu** Gównego

2.9.4 Dokumentacja funkcji składowych

2.9.4.1 void Menu.CreateGame ()

Przejście do nowej sceny - utworzenie rozgrywki

2.9.4.2 void Menu.CreateGameOnOneDevice ()

Przejście do nowej sceny - utworzenie rozgrywki na jednym urządzeniu

2.9.4.3 void Menu.ExitApplication ()

Sprawdzenie platformy i zamknięcie aplikacji

2.9.4.4 void Menu.Info ()

Pokazujemy panel z informacjami

2.9.4.5 void Menu.InfoExit ()

Ukrywamy panel z informacjami

2.9.4.6 void Menu.JoinGame ()

Przejście do nowej sceny - dołączenie do rozgrywkui

2.9.4.7 void Menu.StartTutorial ()

Przejście do nowej sceny - samouczka

2.9.4.8 Dokumentacja dla tej klasy została wygenerowana z pliku:

Menu.cs

2.10 Dokumentacja klasy MenuBoard

Klasu **MenuBoard** Odpowiada za kontrolę planszy w menu głównym
Dziedziczy MonoBehaviour.

2.10.1 Pola danych

float **rotationSpeed**

2.10.2 Opis szczegółowy

Klasu **MenuBoard** Odpowiada za kontrolę planszy w menu głównym

---

2.10.3 Dokumentacja pól

2.10.3.1 float MenuBoard.rotationSpeed

2.10.3.2 Dokumentacja dla tej klasy została wygenerowana z pliku:

MenuBoard.cs

2.11 Dokumentacja klasy Options

Options - Klasa zażądzająca opcjami gry
Dziedziczy MonoBehaviour.

2.11.1 Metody publiczne

void **backToDefaultSettings** ()
void **setGameplayName** (string name)
string **getGameplayName** ()
void **setPawnColor** (PawnsColors color)
PawnsColors **getPawnColor** ()
void **setMusicVolume** (float musicVol)
float **getMusicVolume** ()
void **setSoundVolume** (float soundVol)
float **getSoundVolume** ()
void **setGraphicsQuality** (int gQ)

int **getGraphicsQuality** ()

2.11.2 Pola danych

GameMode **gMode**
float **musicVolume**
Opcje dźwięku
float **soundsVolume**
GameObject **Music**
GameObject **Sounds**
int **graphicsQuality**
Opcje grafiki
int **boardStyle**
int **pawnStyle**
string **gameplayName**
PawnsColors **pawnColor**
const int **gamePort** = 25000

2.11.3 Opis szczegółowy

Options - Klasa zażądzająca opcjami gry

2.11.4 Dokumentacja pól

2.11.4.1 int Options.graphicsQuality

Opcje grafiki

2.11.4.2 float Options.musicVolume

Opcje dźwięku

2.11.4.3 Dokumentacja dla tej klasy została wygenerowana z pliku:

Options.cs

2.12 Dokumentacja klasy Pawn

Klasa Pionka
Dziedziczy MonoBehaviour.

2.12.1 Metody publiczne

void **unHighlightPawn** (GameObject pawn)
Funkcja odswietlająca pionka
void **move** (string fieldId)
Funkcja rusza pionka
void **putOut** ()

2.12.2 Pola danych

GameObject **shadow**

int **pawnID**

Identyfikator pionka

string **fieldID**

Identyfikator pola na którym stoi pionek

PawnState **pState**

Stan pionka

bool **isSelected**

Czy pionek jest zaznaczony

2.12.3 Opis szczegółowy

Klasa Pionka

2.12.4 Dokumentacja funkcji składowych

2.12.4.1 void Pawn.move (string *fieldId*)

Funkcja rusza pionka

Szukamy pozycji na którą postawimy pionek
Przenosimy ten pionek na pozycję dest

2.12.4.2 void Pawn.unHighlightPawn (GameObject *pawn*)

Funkcja odswietlająca pionka

2.12.5 Dokumentacja pól

2.12.5.1 string Pawn.fieldID

Identyfikator pola na którym stoi pionek

2.12.5.2 bool Pawn.isSelected

Czy pionek jest zaznaczony

2.12.5.3 int Pawn.pawnID

Identyfikator pionka

2.12.5.4 PawnState Pawn.pState

Stan pionka

2.12.5.5 Dokumentacja dla tej klasy została wygenerowana z pliku:

gameplay/Pawn.cs

2.13 Dokumentacja klasy Player

Klasa Gracza
Dziedziczy MonoBehaviour.

2.13.1 Metody publiczne

void **createPawns** ()
void **makeMove** (string idField)

Funkcja wykonująca fizyczny ruch

void **makeCapture** (string idOfDestField, int idOfCapturedPawn)

Funkcja wykonująca bicie

void **changeTurn** (**Pawn** pawn2Move)
void **waitForTurn** ()

2.13.2 Pola danych

int **playerID**
GameObject[] **Pawns**
int **pawnToMove**

2.13.3 Opis szczegółowy

Klasa Gracza

2.13.4 Dokumentacja funkcji składowych

2.13.4.1 void Player.changeTurn (Pawn *pawn2Move*)

Trzeba jeszcze odznaczyć pionka po wykonanym ruchu
Zmiana tury
Reszte pól trzeba odznaczyć

2.13.4.2 void Player.createPawns ()

Tworzymy obiekt pionka z prefaba
Dodajemy skrypt dla pionka
Przypisujemy nazwę dla pionka
Skalujemy obiekt
Ustawiamy pionka na polu Przypisujemy ID Pionka dla pola
Dodajemy Box collider

Obracamy pionek o 45 stopni

Przypisujemy pionki dla playera oraz numery id dla pionków, oraz numery id pól na których stoją pionki

Zmieniamy stan pola na którym stawiamy pionka:

Dodajemy tag dla pionka

2.13.4.3 void Player.makeCapture (string *idOfDestField*, int *idOfCapturedPawn*)

Funkcja wykonująca bicie

Ruszamy pionkiem i ustawiamy mu nowe id pola na które ruszył

Zmieniamy status pola z którego ruszyliśmy

Zmieniamy status pola z którego ruszyliśmy

Trzeba zlikwidować pionka wroga podczas likwidacji trzeba także zmienić parametry pola z którego zbity jest pionek oraz parametry bitego pionka

Zerujemy wartość id pionka który stał na polu

Odkładamy pionek na bok

2.13.4.4 void Player.makeMove (string *idField*)

Funkcja wykonująca fizyczny ruch

Ruszamy pionkiem i ustawiamy mu nowe id pola na które ruszył

zmieniamy status pola na którego postawiliśmy pionka

Zmieniamy status pola z którego ruszyliśmy

2.13.4.5 Dokumentacja dla tej klasy została wygenerowana z pliku:

gameplay/Player.cs

2.14 Dokumentacja klasy Tutorial

Klasa samouczka
Dziedziczy MonoBehaviour.

2.14.1 Metody publiczne

void **BackToMenu** ()

*Funkcja powrotu do **Menu** Głównego*

2.14.2 Opis szczegółowy

Klasa samouczka

2.14.3 Dokumentacja funkcji składowych

2.14.3.1 void Tutorial.BackToMenu ()

Funkcja powrotu do **Menu** Głównego

2.14.3.2 Dokumentacja dla tej klasy została wygenerowana z pliku:

Tutorial/Tutorial.cs

3 Podsumowanie

3.1 Aplikacja działa stabilnie i poprawnie oraz spełnia wszystkie wymagania funkcjonalne oraz nie funkcjonalne - rozgrywka jest satysfakcjonująca

3.2 Apliakacja zostałą umieszczona w sklepie Google Play w celu jej dystrybucji

3.3 Aplikacja działa na systemach Android 4.0.3 i nowszych

3.4 Aplikacja została wykonana w środowisku Unity , wykorzystywany był język programowania C#

3.5 Testy zostały przeprowadzone pomyślnie, napotkane błędy usunięte,

3.6 Odpowiednie diagramy i tabele zostały dodane w załączniku w celu zwiększenia ich czytelności

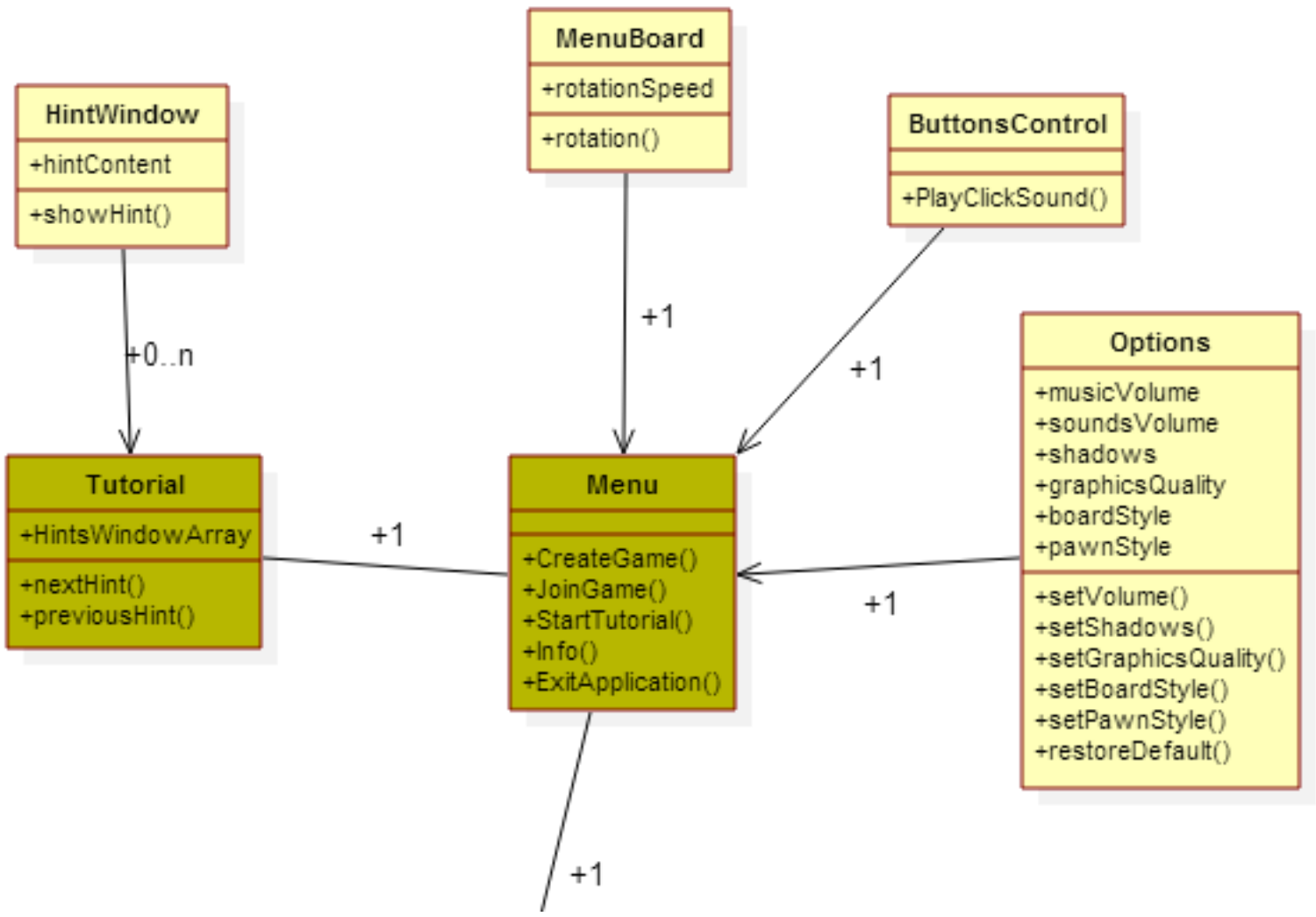
4 Indeks

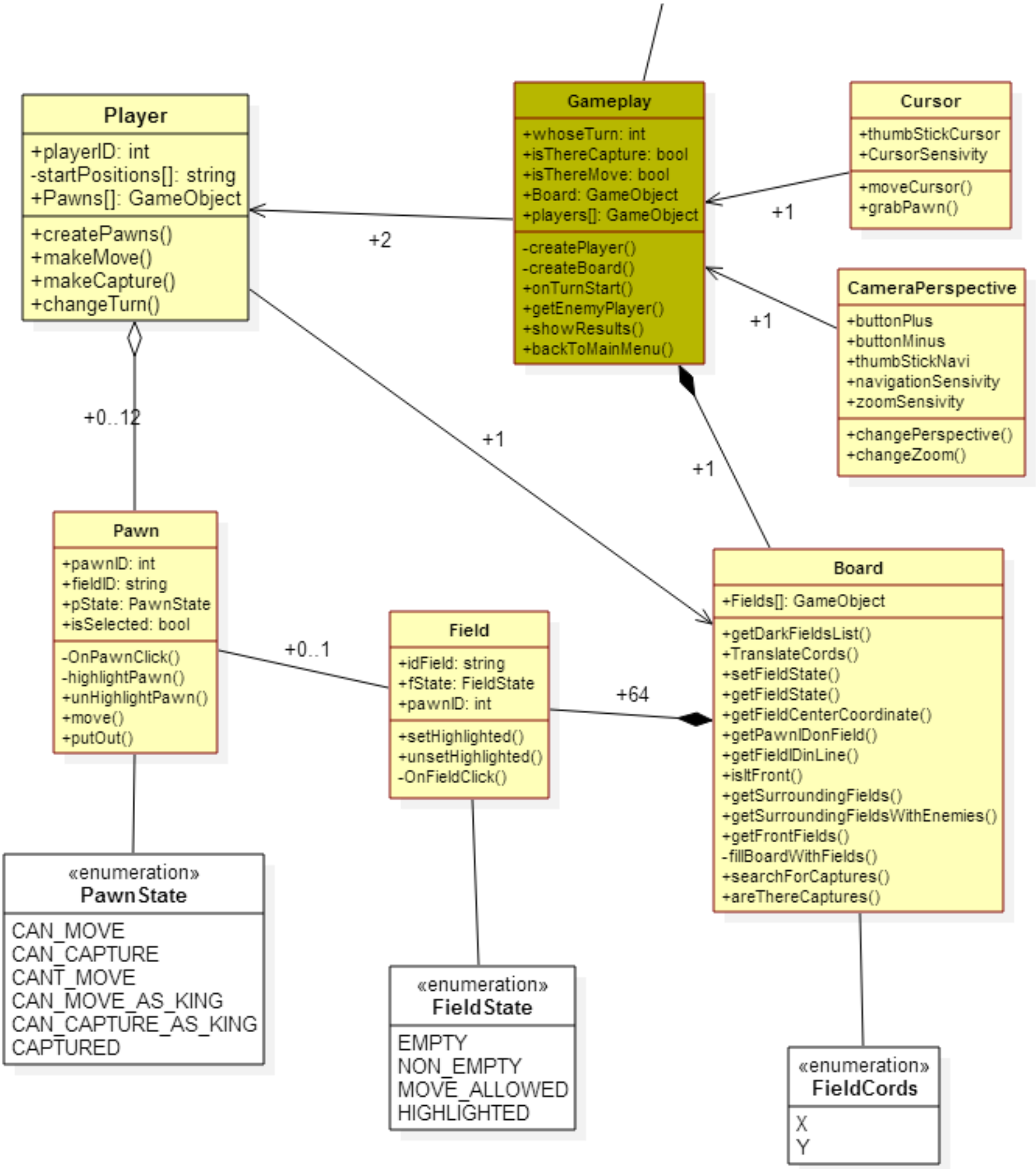
areThereCaptures
Board, 6
backToMainMenu
Gameplay, 12
BackToMenu
Tutorial, 20
Board, 6
areThereCaptures, 6
getDarkFieldsList, 7
getFieldCenterCoordinate, 7
getFieldIDinLine, 7
getFieldState, 7
getPawnIDonField, 7
getSurroundingFields, 7
getSurroundingFieldsWithEnemies, 7
isItFront, 7
searchForCaptures, 7
setFieldState, 8
TranslateCords, 8
ButtonsControl, 8
PlayClickSound, 8
CameraPerspective, 9
setFreeView, 9
setTopView, 9
changeTurn
Gameplay, 12
Player, 18
CreateGame
Menu, 14
CreateGameOnOneDevice
Menu, 14
createPawns
Player, 18
Cursor, 9
ExitApplication
Menu, 14
Field, 10
fState, 10
idField, 10
pawnId, 10
FieldCords, 11
fieldID
Pawn, 17
fState
Field, 10
Gameplay, 11
backToMainMenu, 12
changeTurn, 12
getEnemyPlayer, 12
onTurnStart, 12
panelGameEnd, 13
showResults, 12
whoseTurnID, 13
getDarkFieldsList
Board, 7
getEnemyPlayer
Gameplay, 12
getFieldCenterCoordinate
Board, 7
getFieldIDinLine
Board, 7
getFieldState
Board, 7
getPawnIDonField
Board, 7
getSurroundingFields
Board, 7
getSurroundingFieldsWithEnemies
Board, 7
graphicsQuality
Options, 16
HintWindow, 13
idField
Field, 10
Info
Menu, 14
InfoExit
Menu, 14
isItFront
Board, 7
isSelected
Pawn, 17
JoinGame
Menu, 14
makeCapture
Player, 19
makeMove
Player, 19
Menu, 13
CreateGame, 14
CreateGameOnOneDevice, 14
ExitApplication, 14
Info, 14
InfoExit, 14
JoinGame, 14
StartTutorial, 15
MenuBoard, 15
rotationSpeed, 15
move
Pawn, 17
musicVolume
Options, 16
onTurnStart
Gameplay, 12
Options, 15
graphicsQuality, 16
musicVolume, 16
panelGameEnd
Gameplay, 13
Pawn, 16
fieldID, 17
isSelected, 17
move, 17
pawnID, 17

pState, 18
unHighlightPawn, 17
pawnId
Field, 10
pawnID
Pawn, 17
PlayClickSound
ButtonsControl, 8
Player, 18
changeTurn, 18
createPawns, 18
makeCapture, 19
makeMove, 19
pState
Pawn, 18
rotationSpeed
MenuBoard, 15
searchForCaptures
Board, 7
setFieldState
Board, 8
setFreeView
CameraPerspective, 9
setTopView
CameraPerspective, 9
showResults
Gameplay, 12
StartTutorial
Menu, 15
TranslateCords
Board, 8
Tutorial, 19
BackToMenu, 20
unHighlightPawn
Pawn, 17
whoseTurnID
Gameplay, 13

5 Załączniki

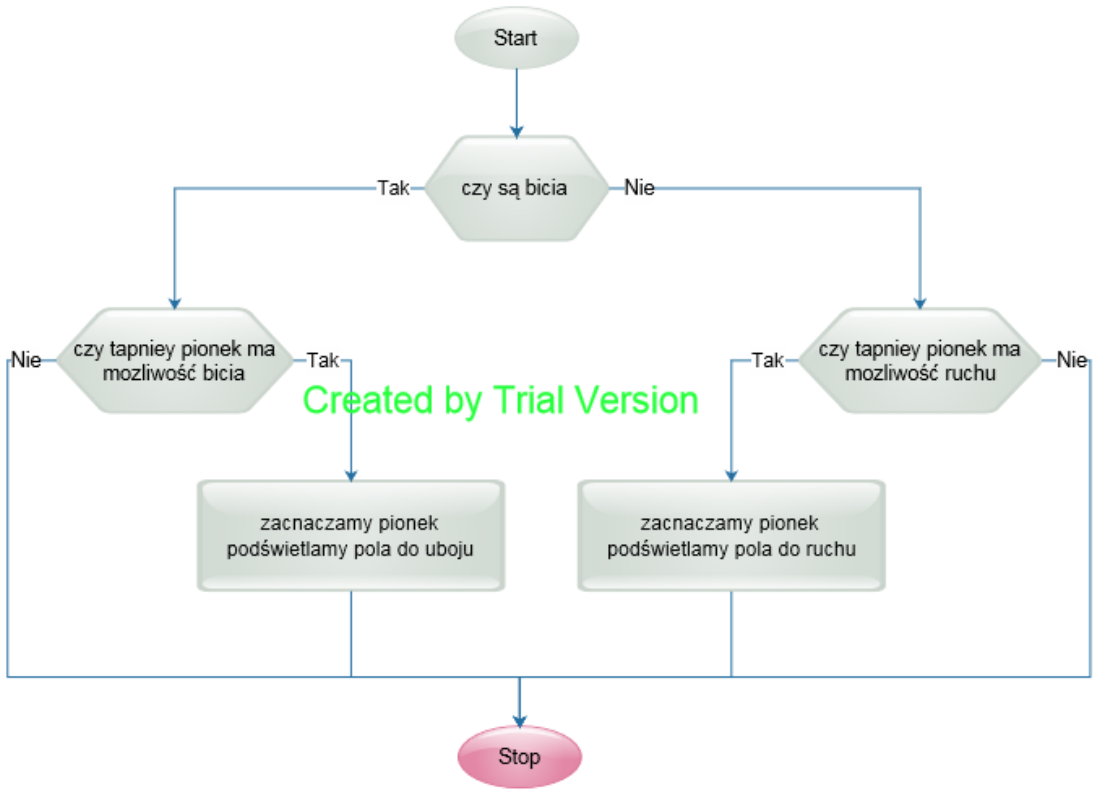
5.1 Diagram Klas



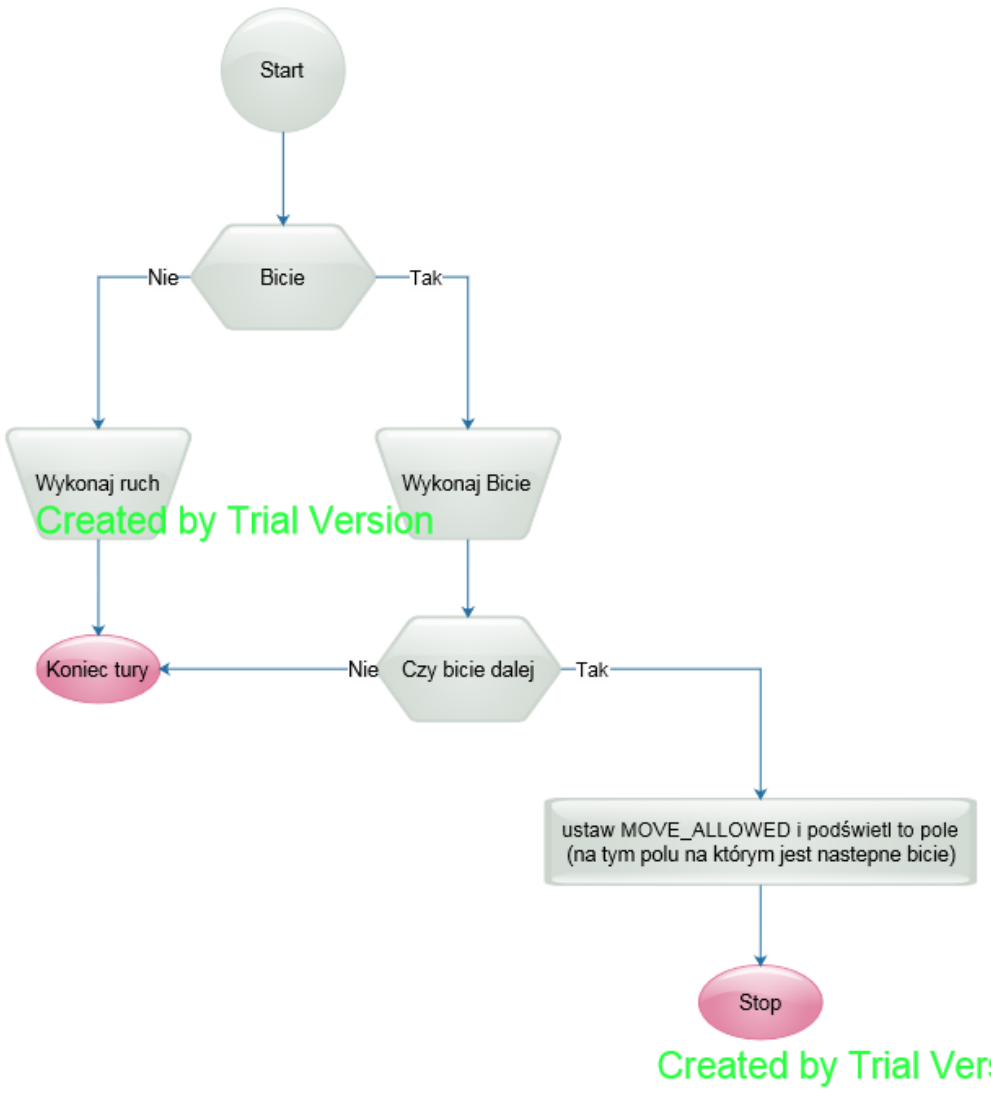

Player
+playerID: int
-startPositions[]: string
+Pawns[]: GameObject
+createPawns()
+makeMove()
+makeCapture()
+changeTurn()
Gameplay
+whoseTurn: int
+isThereCapture: bool
+isThereMove: bool
+Board: GameObject
+players[]: GameObject
-createPlayer()
-createBoard()
+onTurnStart()
+getEnemyPlayer()
+showResults()
+backToMainMenu()
Cursor
+thumbStickCursor
+CursorSensivity
+moveCursor()
+grabPawn()
CameraPerspective
+buttonPlus
+buttonMinus
+thumbStickNavi
+navigationSensivity
+zoomSensivity
+changePerspective()
+changeZoom()
+2
+1
+1
+1
+0..12
+1
Pawn
+pawnID: int
+fieldID: string
+pState: PawnState
+isSelected: bool
-OnPawnClick()
-highlightPawn()
+unHighlightPawn()
+move()
+putOut()
+0..1
Field
+idField: string
+fState: FieldState
+pawnID: int
+setHighlighted()
+unsetHighlighted()
-OnFieldClick()
+64
Board
+Fields[]: GameObject
+getDarkFieldsList()
+TranslateCords()
+setFieldState()
+getFieldState()
+getFieldCenterCoordinate()
+getPawnIDonField()
+getFieldIDinLine()
+isItFront()
+getSurroundingFields()
+getSurroundingFieldsWithEnemies()
+getFrontFields()
-fillBoardWithFields()
+searchForCaptures()
+areThereCaptures()
«enumeration»
PawnState
CAN_MOVE
CAN_CAPTURE
CANT_MOVE
CAN_MOVE_AS_KING
CAN_CAPTURE_AS_KING
CAPTURED
«enumeration»
FieldState
EMPTY
NON_EMPTY
MOVE_ALLOWED
HIGHLIGHTED
«enumeration»
FieldCords
X
Y

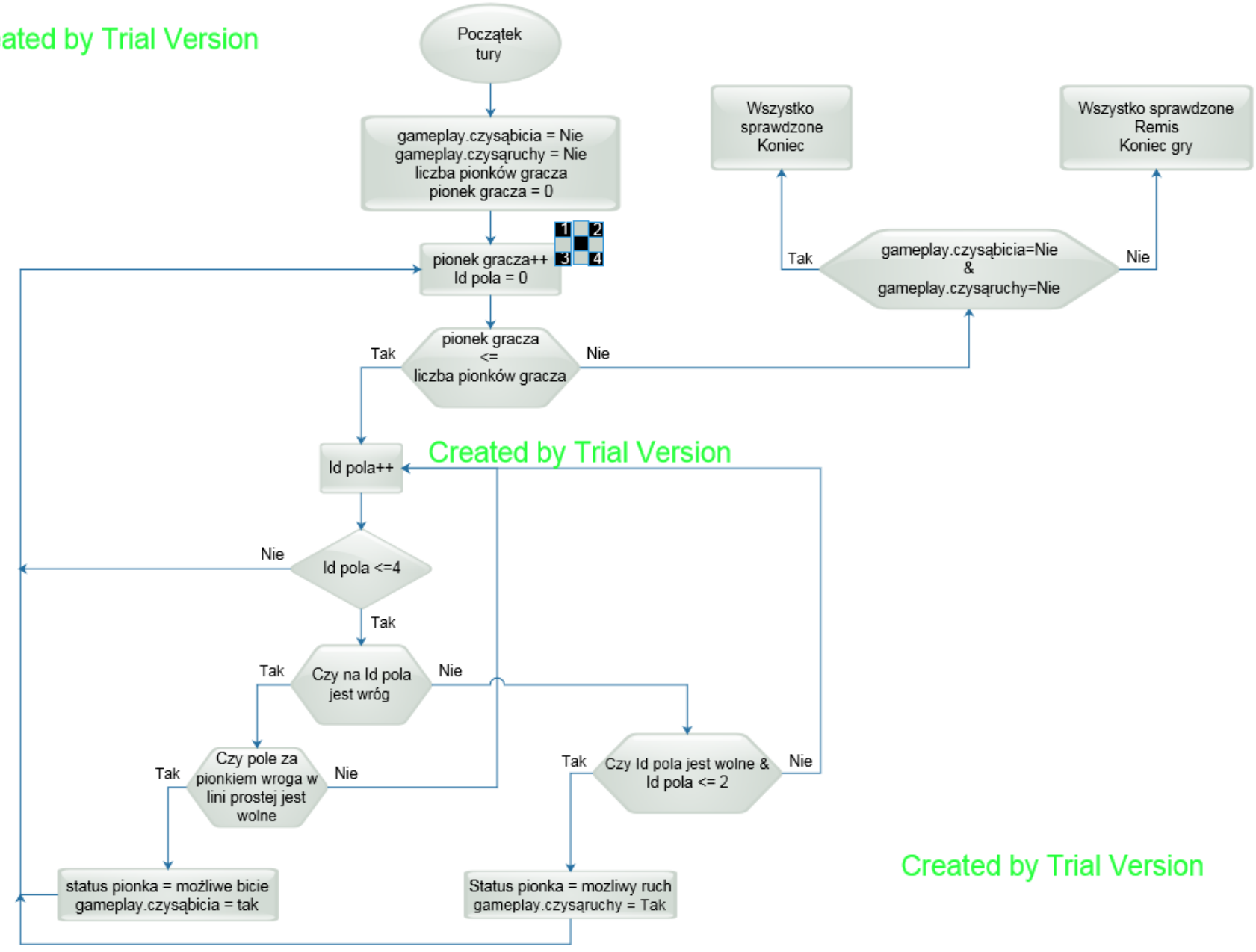
5.2 Schematy blokowe użyte podczas projektowania aplikacji:

Po tapnięciu na pionka



Po tapnięciu na pole





5.3 Diagram Przepływu danych: