Štiri v vrsto na STM32H7

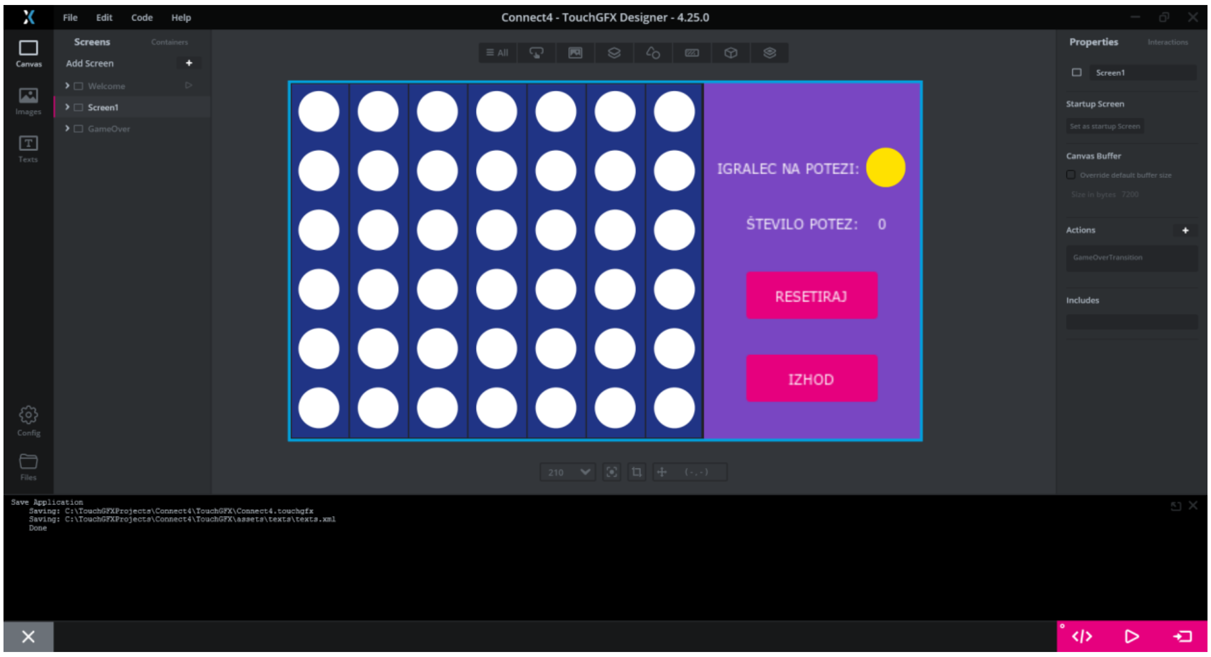
2. domača naloga pri predmetu Organizacija Računalnikov

1. TouchGFX

Projekt je narejen s pomočjo STM-ovega okolja za programiranje grafičnih aplikacij, TouchGFX. V programu se ustvari grafično podobo, za katero TouchGFX generira kodo, logika pa se doda ročno z nadgradnjo generirane kode.

Za svoj projekt sem ustvaril 3 zaslone, ki so prikazani na začetku, koncu in med igro. Med njimi igralec prehaja z gumbi »Začni igro« in »Izhod«.

V glavnem igralnem zaslonu je 7x6 igralno polje žetonov, trenuten status igre in gumbi za resetiranje igre ali izhod na začetni zaslon.



```
Save Application
    Saving: C:\TouchGFXProjects\Connect4\TouchGFX\Connect4.touchgfx
    Saving: C:\TouchGFXProjects\Connect4\TouchGFX\assets\texts\texts.xml
    Done
```

V TouchGFX se lahko tudi definira preproste interakcije z gradniki, pod zavihkom »Interactions«. Tam so definirani stolpci igralnega polja kot gumbi, ki spustijo žeton, in prehodi med zasloni.

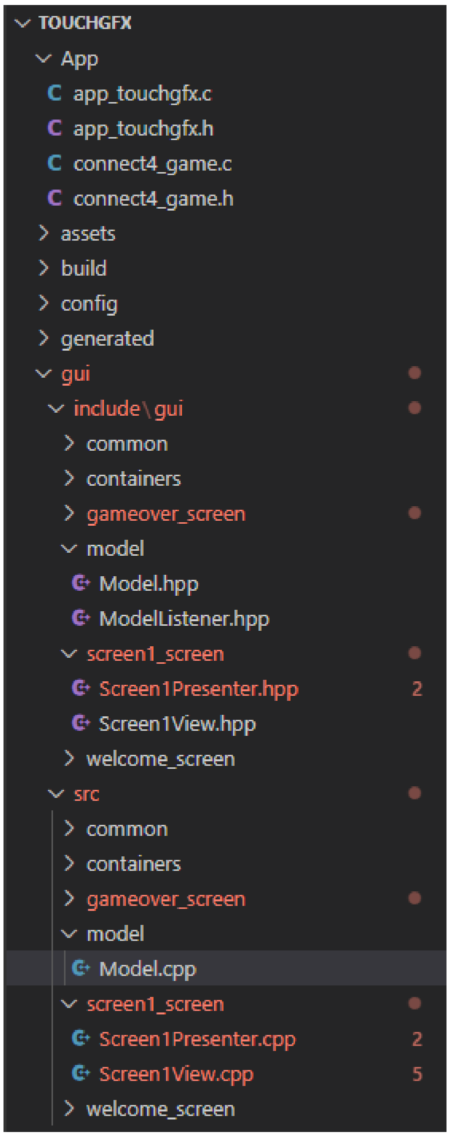
2. Arhitektura aplikacije Model-View-Presenter

Splošna arhitektura TouchGFX aplikacij je Model-View-Presenter, kjer Model predstavlja podatke aplikacije, View grafično predstavitev in Presenter vmesno logiko, ki pošilja signle uporabnika iz View logiki aplikacije v Model, in posodablja View glede na stanje podatkov v Model.

V datotečni strukturi je vsak od teh komponent svoj razred, ki kliče naslednjega in obratno. V View sem implementiral izris igralnega polja, ki se sproži ob spremembi stanja igre. Tu so implementirane tudi visoko-nivojske funkcije, ki jih sproži igralec, in kličejo naprej funkcije od Presenter.

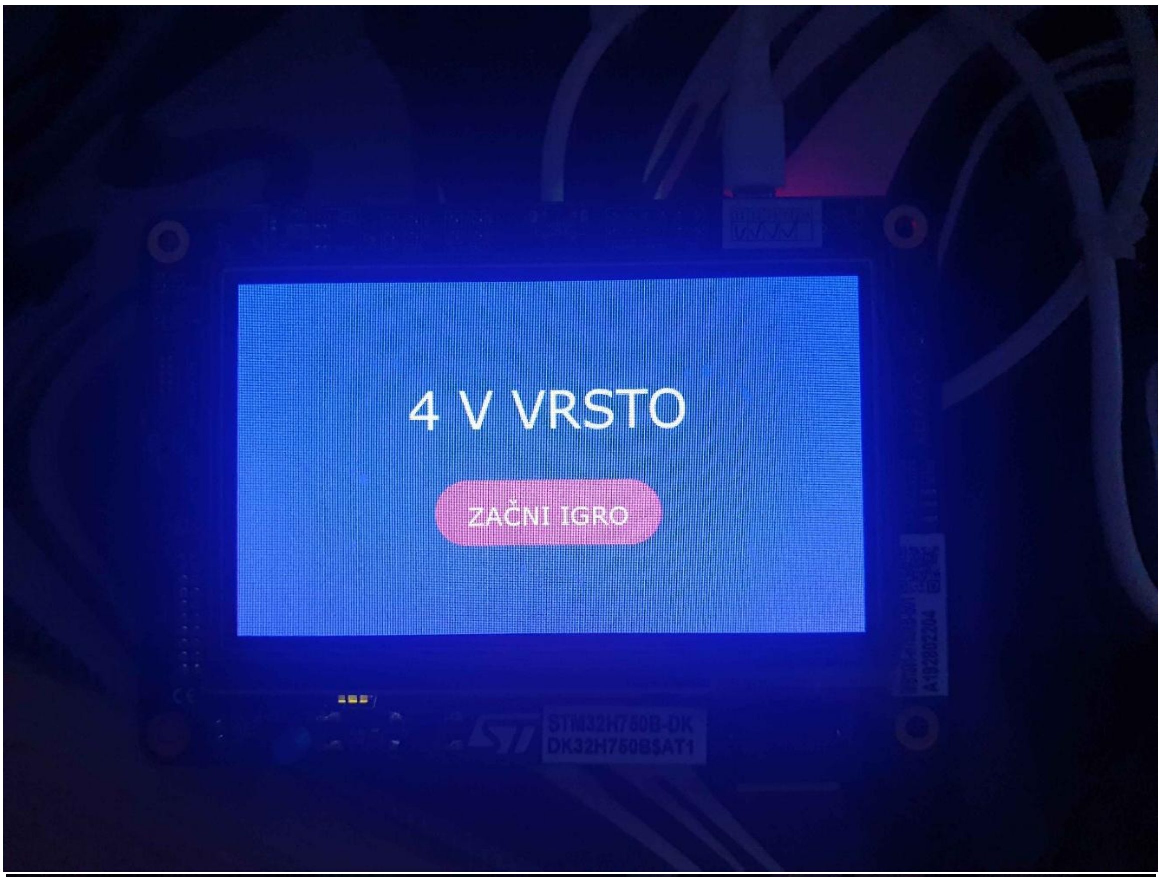
Presenter kliče naprej funkcije od Model, ki spremenijo stanje igre, in sproži izris v View, ko se stanje spremeni.

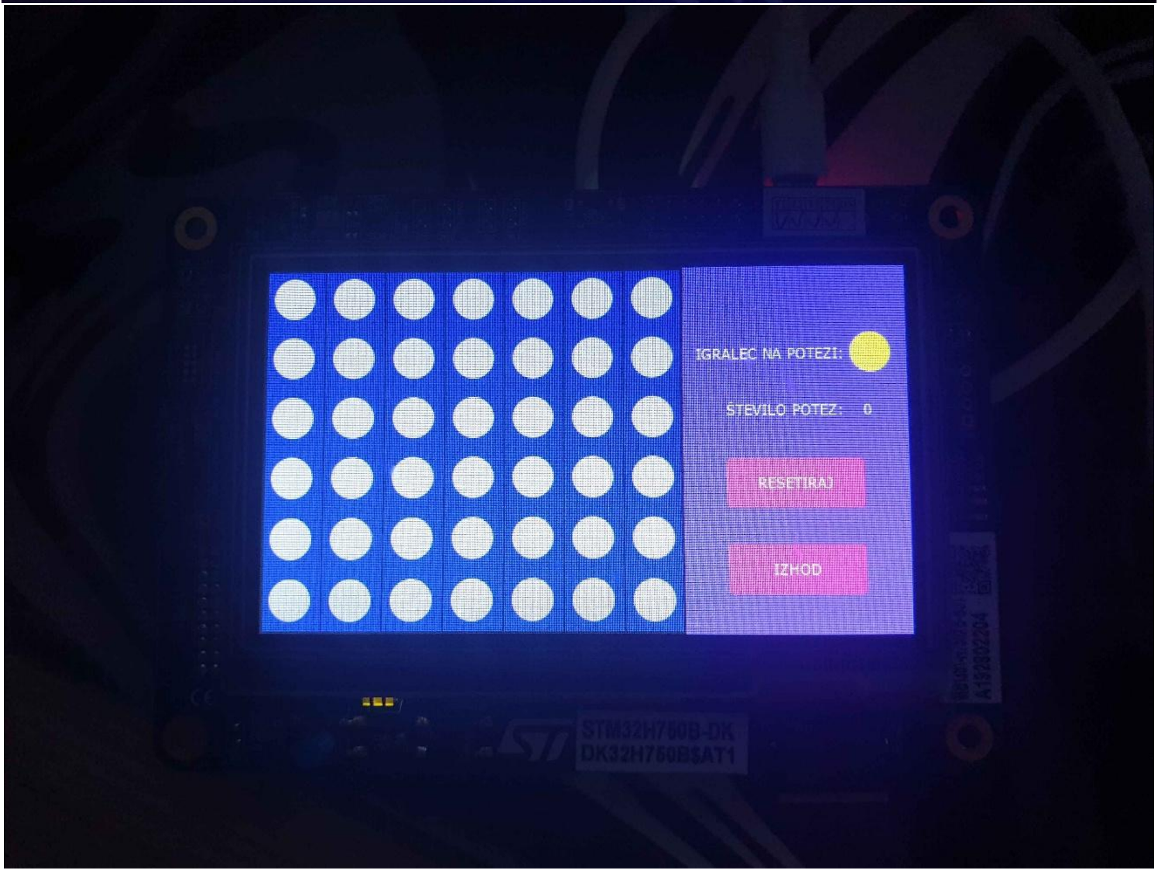
Model je najbolj povezan z logiko aplikacije, ker ji posreduje akcije uporabnika. V projektu je to samo funkcija »dropPiece()«, saj je to edina interakcija uporabnika pri igri Štiri v vrsto. Model kliče funkcije definirane v logiki, v mojem primeru je to »connect4_game.c«, in posreduje signal Presenter, ko se stanje igre spremeni.

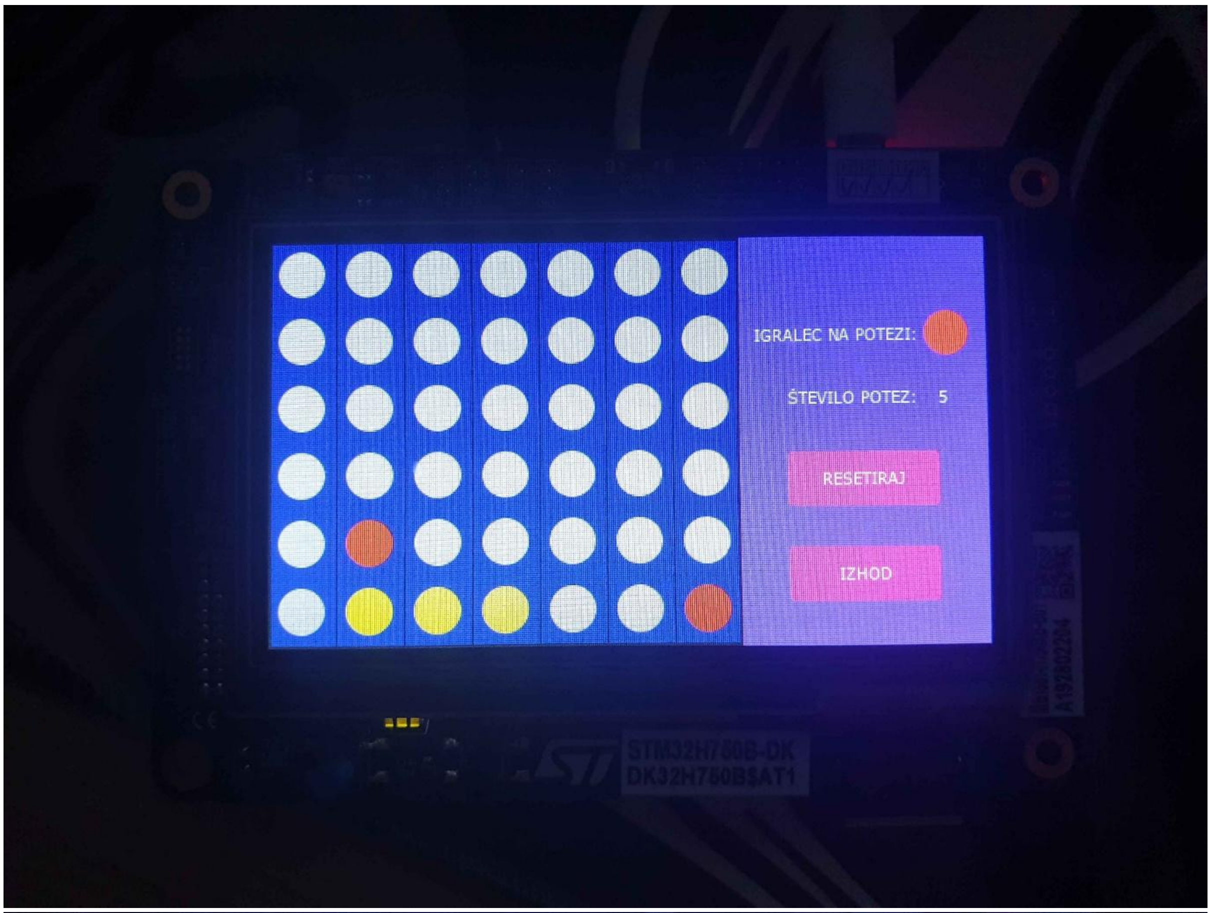


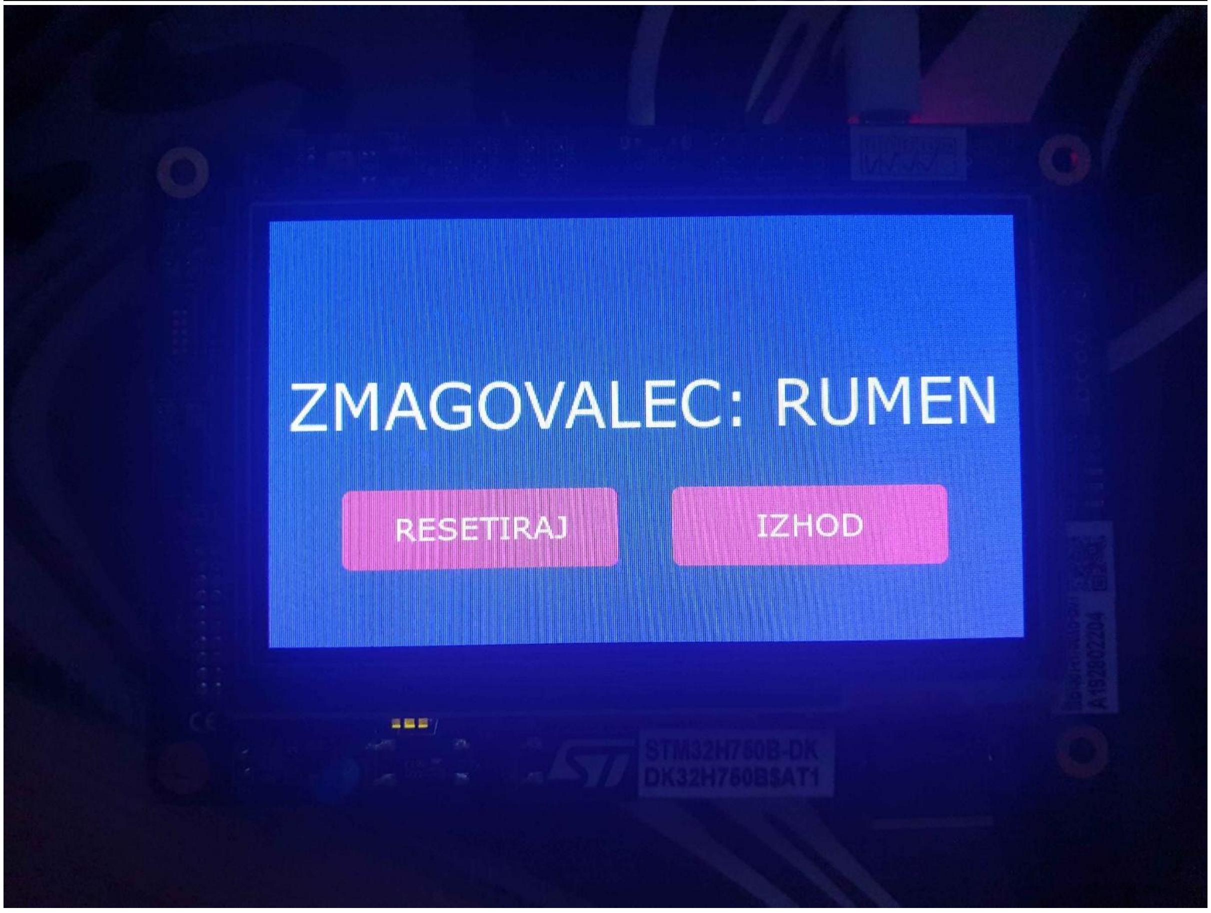
```
TOUCHGFX
  App
    app_touchgfx.c
    app_touchgfx.h
    connect4_game.c
    connect4_game.h
  assets
  build
  config
  generated
  gui
    include\gui
      common
      containers
      gameover_screen
      model
        Model.hpp
        ModelListener.hpp
      screen1_screen
        Screen1Presenter.hpp        2
        Screen1View.hpp
      welcome_screen
    src
      common
      containers
      gameover_screen
      model
        Model.cpp
      screen1_screen
        Screen1Presenter.cpp        2
        Screen1View.cpp             5
      welcome_screen
```

3. Rezultat


4 V VRSTO
ZAČNI IGRO
STM32H750B-DK
DK32H750B$AT1


IGRALEC NA POTEZI:
STEVILO POTEZ: 0
RESETIRAJ
IZHOD
STM32H750B-DK
DK32H750B$AT1


IGRALEC NA POTEZI:
ŠTEVILO POTEZ: 5
RESETIRAJ
IZHOD
STM32H750B-DK
DK32H750B$AT1


ZMAGOVALEC: RUMEN
RESETIRAJ
IZHOD
STM32H750B-DK
DK32H750B$AT1

4. Priloge

connect4_game.h:

```
#ifndef CONNECT4_GAME_H
#define CONNECT4_GAME_H

#include <stdint.h>
#include <stdbool.h>

#ifdef __cplusplus
extern "C" {
#endif

#define BOARD_ROWS 6
#define BOARD_COLS 7

typedef enum {
    PLAYER_NONE = 0,
    PLAYER_YELLOW = 1,
    PLAYER_RED = 2
} Player_t;

typedef enum {
    GAME_STATE_PLAYING,
    GAME_STATE_DRAW,
    GAME_STATE_WIN
} GameState_t;

typedef struct {
    Player_t board[BOARD_COLS][BOARD_ROWS];
    Player_t currentPlayer;
    GameState_t gameState;

    Player_t winner;

    uint8_t movesCount;

} Connect4Game_t;


void Connect4Game_InitInstance();

void Connect4Game_Init(Connect4Game_t* game);

bool Connect4Game_DropPiece(Connect4Game_t* game, uint8_t
column);

bool Connect4Game_CheckWin(Connect4Game_t* game, uint8_t
lastCol, uint8_t lastRow);

bool Connect4Game_CheckDraw(Connect4Game_t* game);

void Connect4Game_Reset(Connect4Game_t* game);

Connect4Game_t* Connect4Game_GetInstance();


bool Connect4Game_IsColumnFull(Connect4Game_t* game, uint8_t
column);

Player_t Connect4Game_GetPieceAt(Connect4Game_t* game, uint8_t
column, uint8_t row);


#ifdef __cplusplus

}

#endif

#endif
```

connect4_game.c:

```
#include "connect4_game.h"

static Connect4Game_t gameInstance;

Connect4Game_t* Connect4Game_GetInstance() {
    return &gameInstance;
}

void Connect4Game_InitInstance() {
    Connect4Game_Init(&gameInstance);
}

void Connect4Game_Init(Connect4Game_t* game) {
    for (uint8_t col = 0; col < BOARD_COLS; col++) {
        for (uint8_t row = 0; row < BOARD_ROWS; row++) {
            game->board[col][row] = PLAYER_NONE;
        }
    }

    game->currentPlayer = PLAYER_YELLOW; // Začetni igralec
rumen
    game->gameState = GAME_STATE_PLAYING;
    game->winner = PLAYER_NONE;
    game->movesCount = 0;
}

bool Connect4Game_DropPiece(Connect4Game_t* game, uint8_t
column) {
    if (Connect4Game_IsColumnFull(game, column)) {
        return false;
    }
    if (column >= BOARD_COLS) {
        return false;
    }

    for (int8_t row = BOARD_ROWS - 1; row >= 0; row--) {
        if (game->board[column][row] == PLAYER_NONE) {
            game->board[column][row] = game->currentPlayer;
            game->movesCount++;

            if (Connect4Game_CheckWin(game, column, row)) {
                game->winner = game->currentPlayer;
                game->gameState = GAME_STATE_WIN;
            } else if (Connect4Game_CheckDraw(game)) {
                game->winner = PLAYER_NONE;
                game->gameState = GAME_STATE_DRAW;
            } else { // Če ni zmagovalca ali izenačenja
zamenjaj igralca
                game->currentPlayer = (game->currentPlayer ==
PLAYER_YELLOW) ? PLAYER_RED : PLAYER_YELLOW;
            }
            return true;
        }
    }

    return false; // "nedosegljivo"
}

bool Connect4Game_IsColumnFull(Connect4Game_t* game, uint8_t
column) {
    return game->board[column][0] != PLAYER_NONE; // najvišja
vrstica
```

```
}


bool Connect4Game_CheckDraw(Connect4Game_t* game) {
    return game->movesCount == (BOARD_ROWS * BOARD_COLS); //
Potez je toliko kot je polj, tabela je polna in ni zmagovalca
}


bool Connect4Game_CheckWin(Connect4Game_t* game, uint8_t
lastCol, uint8_t lastRow) {
    uint8_t pieceCount;


    // Navpično
    pieceCount = 1; // Že postavljen
    for (int8_t row = lastRow + 1; row < BOARD_ROWS; row++) {
        if (game->board[lastCol][row] != game->currentPlayer)
break;
        pieceCount++;
    }
    for (int8_t row = lastRow - 1; row >= 0; row--) {
        if (game->board[lastCol][row] != game->currentPlayer)
break;
        pieceCount++;
    }
    if (pieceCount >= 4) return true;


    // Vodoravno
    pieceCount = 1;
    for (int8_t col = lastCol + 1; col < BOARD_COLS; col++) {
        if (game->board[col][lastRow] != game->currentPlayer)
break;
        pieceCount++;
    }
    for (int8_t col = lastCol - 1; col >= 0; col--) {
        if (game->board[col][lastRow] != game->currentPlayer)
break;
        pieceCount++;
    }
    if (pieceCount >= 4) return true;

    // Diagonalno (NW->SE)
    pieceCount = 1;
    for (int8_t col = lastCol + 1, row = lastRow + 1; col <
BOARD_COLS && row < BOARD_ROWS; col++, row++) {
        if (game->board[col][row] != game->currentPlayer)
break;
        pieceCount++;
    }
    for (int8_t col = lastCol - 1, row = lastRow - 1; col >= 0
&& row >= 0; col--, row--) {
        if (game->board[col][row] != game->currentPlayer)
break;
        pieceCount++;
    }
    if (pieceCount >= 4) return true;

    // Diagonalno (SW->NE)
    pieceCount = 1;
    for (int8_t col = lastCol - 1, row = lastRow + 1; col >= 0
&& row < BOARD_ROWS; col--, row++) {
        if (game->board[col][row] != game->currentPlayer)
break;
        pieceCount++;
    }
    for (int8_t col = lastCol + 1, row = lastRow - 1; col <
BOARD_COLS && row >= 0; col++, row--) {
        if (game->board[col][row] != game->currentPlayer)
break;
        pieceCount++;
    }
    if (pieceCount >= 4) return true;


    return false;
}


Player_t Connect4Game_GetPieceAt(Connect4Game_t* game, uint8_t
column, uint8_t row) {
    if (column < BOARD_COLS && row < BOARD_ROWS) {
        return game->board[column][row];
    }
    return PLAYER_NONE;
}


void Connect4Game_Reset(Connect4Game_t* game) {
    Connect4Game_Init(game);
}
```