SoKOTban

Dokumentacja techniczna

Gniewomir Gaudyn

Spis treści

1 Wstęp **1**

2 Podział modułów **2**
2.1 Warstwa modelu 2
2.2 Warstwa viewmodelu 2
2.3 Warstwa widoku 2

3 Opis modułów warstwy modelu **2**
3.1 Board 2
3.2 ActionList 4

4 Opis modułów warstwy viewmodelu **5**
4.1 BoardLogic 5
4.2 BoardOperations 6

5 Opis modułów warstwy widoku **7**
5.1 MainWindow 7
5.2 GameGrid 7
5.3 GameToolbar 8
5.4 EditorToolbar 9
5.5 EditorMenu 9
5.6 MainGame 10
5.7 MainEditor 10

1 Wstęp

SoKOTban to projekt stworzony na potrzeby realizacji przedmiotu "Wstępu do programowania w języku C". Jego głównym celem było napisanie jednoosobowej gry Śokobanź użyciem biblioteki GTK+ 3. Sama gra została poszerzona o edytor poziomów, które są główną częścią tej gry, w związku z tym bez edytora poziomów byłoby to niepełne doświadczenie dla użytkownika.
Od tego momentu program do rozwiązywania łamigłówek będzie nazywany grą, a program do edycji plansz edytorem.

2 Podział modułów

Projekt został podzielony na moduły biorąc pod uwagę wzorzec MVVM, jednak przez specyfikę języka C nie można powiedzieć, że spełnia on główne założenia tego wzorca.

2.1 Warstwa modelu

Moduły w tej warstwie zajmują się przechowywaniem danych użytkownika, czyli planszy do gry (moduł Board) oraz kolejki czynności (moduł ActionList). Oba moduły są wykorzystywane przez grę i edytor.

2.2 Warstwa viewmodelu

Moduły w tej warstwie zajmują się komunikacją między warstwą widoku i warstwą modelu. Po edycji danych w warstwie modelu następuje aktualizacja widoku przy pomocy nowych danych. Dla gry jest to moduł BoardLogic, a dla edytora plansz BoardOperations.

2.3 Warstwa widoku

Zajmuje się tworzeniem interfejsu, jego aktualizacją i reagowaniem na czynności użytkownika. Jedyny moduł z tej warstwy, który jest używany przez oba programy to MainWindow.

3 Opis modułów warstwy modelu

3.1 Board

Jest to moduł odpowiedzialny za implementację plansz do gry.
Plansza definiujemy za pomocą jej długości, szerokości, prostokątnej tablicy pól, ścieżki do pliku oraz najwyższemu wynikowi ruchów i przesunięć.

Pola tablicy:

- EMPTY – Puste pole, domyślne wypełnienie tablicy
- WALK – Pole podłogi
- BRICK – Pole ściany
- PLAYER – Pole startowe gracza
- BOX – Pole startowe dla pudełek
- BUTTON –Pole docelowe dla pudełek

Typy pól są trzymane w pamięci jako enum tile.

Board boardInit(int sizeX, int sizeY)
Tworzy i zwraca pustą planszę o podanych wymiarach. Jeżeli operacja się nie powiodła zwraca NULL.

Board boardDestroy(Board b)
Zwalnia pamięć zajętą przez planszę, zwraca NULL.

bool saveBoard(Board b, char *filePath)
Zapisuje podaną planszę do pliku o podanej ścieżce. Zwraca true jeśli udało się zapisać planszę, false w przeciwnym wypadku.

Board loadBoard(char * filePath)
Wczytuje planszę z pliku o podanej ścieżce. Zwraca planszę jeśli udało się wczytać, NULL w przeciwnym wypadku.

int getBoardSizeX(Board b)
Zwraca szerokość planszy.

int getBoardSizeY(Board b)
Zwraca wysokość planszy.

void setBoardFilepath(Board b, char *filePath)
Ustawia ścieżkę podanej planszy.

char *getBoardFilepath(Board b)
Zwraca ścieżkę podanej planszy.

enum tile getBoardValueAt(Board b, int posX, int posY)
Zwraca typ płytki na współrzędnych (posX, posY) podanej planszy.

void setBoardValueAt(Board b, int posX, int posY, enum tile value)
Ustawia typ płytki na współrzędnych (posX, posY) podanej planszy na value

char *getBoardFilename(Board b)
Zwraca nazwę pliku podanej planszy (różni się od pełnej ścieżki pliku).

int getBoardHiMoves(Board b)
Zwraca najwyższy wynik ruchów na danej planszy.

void setBoardHiMoves(Board b, int value)
Ustawia najwyższy wynik ruchów na value na podanej planszy.

int getBoardHiPushes(Board b)
Zwraca najwyższy wynik przesunięć na danej planszy.

void setBoardHiPushes(Board b, int value)
Ustawia najwyższy wynik przesunięć na value na podanej planszy.

int verifyBoard(Board b)
Waliduje planszę. Zwraca 0 jeżeli plansza jest prawidłowa, PLAYER jeżeli jest nieprawidłowa liczba miejsc startowych na planszy, BOX jeżeli pudełek jest więcej niż miejsc docelowych, BUTTON jeżeli miejsc docelowych jest więcej niż pudełek.

Board copyBoard(Board b)
Zwraca kopię podanej planszy zapisaną w innym miejscu pamięci lub NULL jeśli operacja się nie powiodła.

void printBoard(Board b)
Wypisuje tablicę płytek podanej planszy na standardowe wyjście.

3.2 ActionList

Moduł zajmujący się kolejką czynności użytkownika, przechodzi po dwukierunkowej liście wiązanej kolejnych zmian w planszy.

gboolean addActionToList(Board b)
Dodaje planszę na koniec listy. Zwraca TRUE jeżeli operacja się udała, FALSE w przeciwnym wypadku. Kiedy obecna plansza nie był ostatnią, usuwa wszystkie dalsze plansze.

Board undoActionFromList(void)
Zwraca planszę, która jest poprzednikiem obecnej planszy lub NULL jeśli obecna plansza jest pierwsza na liście.

Board redoActionFromList()
Zwraca planszę, która jest następnikiem obecnej planszy lub NULL jeśli obecna plansza jest ostatnia na liście.

gboolean isCurrActionLast()
Zwraca TRUE jeśli obecna plansza jest ostatnia w liście, FALSE w przeciwnym przypadku.

gboolean isCurrActionFirst()
Zwraca TRUE jeśli obecna plansza jest pierwsza w liście, FALSE w przeciwnym przypadku.

void clearActionList(void)
Usuwa wszystkie elementy z listy plansz.

void clearActionListFromCurr(void)
Usuwa wszystkie elementy z listy plansz, które są następnikami obecnej planszy.

4 Opis modułów warstwy viewmodelu

4.1 BoardLogic

Jest to moduł odpowiedzialny za komunikację między interfejsem gry oraz warstwą modelu. Posiada on dwie plansze: główną oraz aktywną kopię, w której jest zapisywany aktualny stan gry.

void initLogic(char *filePath)
Inicjalizuje logikę dla planszy o podanej ścieżce.

void prepareBoard(Board b)
Znajduje pozycję gracza na planszy oraz pól docelowych na pudełka, które potem zapisuje na liście.

void revertBoard(void)
Przywraca aktywną planszę do stanu początkowego.

gboolean isPuzzleSolved(void)
Zwraca TRUE jeśli aktywna plansza została ukończona, FALSE w przeciwnym wypadku.

void puzzleSolved(void)
Finalizuje rozgrywkę oraz wyświetla okno informujące o ukończeniu gry.

gboolean undoLastAction(void)
Cofa ostatni ruch na aktywnej planszy.

gboolean redoLastAction(void)
Powtarza ostatnio cofnięty ruch na aktywnej planszy.

gboolean saveGameState(char *filePath)
Zapisuje stan aktywnej planszy do pliku o podanej ścieżce.

gboolean loadGameState(char *filePath)
Wczytuje zapis z podanej ścieżki do aktywnej planszy.

Board getCurrentBoard(void)
Zwraca aktywną planszę.

Board getMainBoard(void)
Zwraca główną planszę.

void updatePlayerPos(void)
Aktualizuje pozycję gracza po cofnięciu lub powtórzeniu ruchu.

4.2 BoardOperations

Moduł odpowiedzialny za komunikację między interfejsem edytora oraz warstwą modelu.

gboolean createNewBoard(int sizeX, int sizeY)
Tworzy nową, pustą planszę i aktualizuje interfejs. Zwraca TRUE jeżeli operacja się powiodła, FALSE w przeciwnym wypadku.

gboolean openBoardFromFile(char * filepath)
Otwiera planszę z pliku o podanej ścieżce i aktualizuje interfejs. Zwraca TRUE jeżeli operacja się powiodła, FALSE w przeciwnym wypadku.

gboolean saveBoardToFile(char *filepath)
Zapisuje aktywną planszę do pliku. Zwraca TRUE jeżeli operacja się powiodła, FALSE w przeciwnym wypadku.

gboolean undoBoardState(void)
Cofa ostatnią zmianę planszy wykonaną przez użytkownika. Zwraca TRUE jeżeli operacja się powiodła, FALSE w przeciwnym wypadku.

gboolean redoBoardState(void)
Powtarza ostatnio cofniętą zmianę planszy wykonaną przez użytkownika. Zwraca TRUE jeżeli operacja się powiodła, FALSE w przeciwnym wypadku.

gboolean addStateToList(void)
Dodaje aktywną planszę do kolejki czynności użytkownika i aktualizuje menu Edycji. Zwraca TRUE jeżeli operacja się powiodła, FALSE w przeciwnym wypadku.

void changeToBoardTitle(gboolean edited)
Ustawia tytuł głównego okna na nazwę aktywnej planszy.

gboolean fillBoard(GtkWidget * sender, GdkEvent * event, gpointer data)
Ustawia typ płytki na polu kliknięty przez użytkownika aktualnie wybranym typem płytki.

void updateBoard(void)
Aktualizuje wszystkie pola planszy w interfejsie.

void removeWidget(GtkWidget * widget, gpointer data)
Usuwa podany widget.

void changeImageFile(GtkWidget * image, char * file)
Ustawia obraz w podanym widget'cie na plik o podanej ścieżce.

gboolean isBoardEdited(void)
Zwraca TRUE jeżeli plansza została edytowana, FALSE w przeciwnym przypadku.

5 Opis modułów warstwy widoku

5.1 MainWindow

Moduł zajmujący się obsługą głównego okna gry i edytora.

GtkWidget *initMainWindow(void)
Inicjalizuje główne okno programu i zwraca je.

GtkWidget *getMainWindow(void)
Zwraca główne okno programu.

void setMainWindowTitle(char * title)
Ustawia tytuł głównego okna.

gint showEditionAlertDialog(void)
Wyświetla dialog o błędzie edycji planszy.

void showErrorDialog(const char * message)
Wyświetla dialog o błędzie i podanej wiadomości.

5.2 GameGrid

Moduł zajmujący się obsługą głównego grida planszy w grze.

GtkWidget *initBoardGrid(void)
Inicjalizuje głównego grida i zwraca go.

gboolean mainBoardDraw(GtkWidget *widget, cairo_t *cr, gpointer data)
Ustawia kolor tła pod głównym gridem na czarny.

void loadBoardGrid(Board board)
Wczytuje typy płytek dla poszczególnych pól planszy do grida.

void updateBoardGrid(Board board)
Aktualizuje całego grida dla planszy.

void updateGridTile(int posX, int posY, int value)
Aktualizuje miejsce o współrzędnych (posX, posY) płytką o wartości value w gridzie.

int setTileSize(int newSize, Board board)
Ustawia wielkość płytki w gridzie i aktualizuje go. Dla newSize równego 1 powiększa rozmiar o jeden, dla równego -1 zmniejsza rozmiar o jeden piksel

void removeWidget(GtkWidget * widget, gpointer data)
Usuwa podany widget.

void changeImageFile(GtkWidget * image, char * file)
Ustawia obraz w podanym widget'cie na plik o podanej ścieżce.

5.3 GameToolbar

Moduł zajmujący się obsługą paska narzędziowego w grze.

GtkWidget *initMainToolbar(void)
Inicjalizuje i zwraca główny pasek narzędziowy w grze.

void loadNewBoard(GtkWidget *widget, gpointer data)
Wyświetla dialog o wyborze planszy i wczytuje ją do pamięci.

void loadBoardState(GtkWidget *widget, gpointer data)
Wyświetla dialog o wyborze zapisu i wczytuje planszę z zapisu do pamięci.

void saveBoardStateAs(GtkWidget *widget, gpointer data)
Zapisuje obecny stan planszy.

void undoBoardCallback(GtkWidget *widget, gpointer data)
Cofa ostatnią czynność użytkownika.

void redoBoardCallback(GtkWidget *widget, gpointer data)
Powtarza ostatnie cofnięcie użytkownika.

void resetBoard(GtkWidget *widget, gpointer data)
Przywraca planszę do stanu początkowego.

void updateEditButtonsState(gboolean undoState, gboolean redoState)
Ustawia sensitivity przycisków do cofania i powtarzania czynności według parametrów.

void changeMainGridSize(GtkWidget *widget, gpointer data)
Zmienia wielkość głównej planszy.

5.4 EditorToolbar

Moduł zajmujący się paskiem wyboru płytki w edytorze.

GtkWidget *initToolbar(void)
Inicjalizuje pasek z wyborem aktywnej płytki i zwraca go.

void setCurrentTile(GtkWidget * widget, gpointer tile)
Ustawia nazwę aktywnej płyki w pasku narzędziowym.

5.5 EditorMenu

Moduł zajmujący się paskiem narzędziowym w edytorze

GtkWidget * prepareMainWindowMenu(void)
Inicjalizuje i zwraca główny pasek narzędziowy dla edytora.

void showNewBoardDialog(GtkWidget * widget, gpointer data)
Wyświetla dialog z wyborem wielkości nowej, pustej planszy.

void closeDialogWindow(GtkWidget * widget, gpointer data)
Zamyka dialog z wyborem nowej planszy.

void newBoardCallback(GtkWidget * widget, gpointer data)
Tworzy nową planszę o wymiarach podanych w dialogu nowej planszy.

void showOpenBoardDialog(GtkWidget *widget, gpointer data)
Wyświetla dialog z otwieraniem utworzonej planszy.

void saveBoardWithDialog(GtkWidget *widget, gpointer data)
Wyświetla dialog z zapisywaniem aktualnej planszy.

void saveBoardWithoutDialog(GtkWidget *widget, gpointer data)
Zapisuje planszę bez wyświetlania dialogu.

void saveBoardCallback(GtkWidget *widget, gpointer data)
Zapisuje planszę bez dialogu jeśli jest to możliwe, w przeciwnym wypadku wyświetla dialog.

GtkFileChooserConfirmation confirmOverwriteCallback(
GtkFileChooser *chooser, gpointer data)
Obsługuje nadpisywanie plików w dialogu zapisu planszy.

void setSavesEnabled(gboolean sensitivity)
Ustawia sensitivity przycisków do zapisu planszy.

void showInvalidBoardDialog(int code,
void (*saveFunc)(GtkWidget*,gpointer))
Wyświetla dialog o nieprawidłowej walidacji planszy.

void setGridScale(GtkWidget * widget, gpointer data)
Obsługuje zmianę wielkości rozmiaru planszy.

void undoButtonCallback(GtkWidget *widget, gpointer data)
Obsługuje cofnięcie ostatniej czynności użytkownika.

void redoButtonCallback(GtkWidget *widget, gpointer data)
Obsługuje powtórzenie czynności użytkownika.

void updateEditButtons(gboolean undoState, gboolean redoState)
Ustawia sensitivity przycisków do edycji.

5.6 MainGame

Główny moduł gry. Poza inicjalizacją całego interfejsu i wyświetleniem początkowego okna dialogowego zajmuje się przechwytywaniem zdarzeń kliknięcia klawiszy na klawiaturze.

gboolean keypressCallback(GtkWidget *widget, GdkEventKey *event,
GtkWidget *mainBoardGrid)
Odpowiada na kliknięcia klawiszy na klawiaturze.

5.7 MainEditor

Główny moduł edytora. Poza inicjalizacją interfesju tworzy głównego grida i wyświetla okno dialogowe kiedy plansza ma niezapisane zmiany.

void addActionCallback(GtkWidget *widget, gpointer data)

Próbuje dodać ostatnio zakończoną czynność użytkownika do kolejki.

gboolean shouldCloseWindow(GtkWidget *widget, GdkEvent *event, gpointer data)

Wyświetla okno dialogowe przed zamknięciem programu kiedy aktywna plansza ma niezapisane zmiany lub zamyka program w przeciwnym wypadku.