МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное образовательное учреждение высшего образования
«Дальневосточный федеральный университет»
(ДВФУ)

ИНСТИТУТ МАТЕМАТИКИ И КОМПЬЮТЕРНЫХ ТЕХНОЛОГИЙ (ШКОЛА)

Департамент программной инженерии и искусственного интеллекта

Поляков Артём Викторович

Разработка игры “Пятнашки”

ОТЧЕТ
по дисциплине «Учебно-технологическая (проектно-технологическая) практика» по образовательной программе подготовки бакалавров по направлению 09.03.04 «Программная инженерия» профиль «Программная инженерия»

Студент группы Б9122-09.03.04
_____ А.В. Поляков
(подпись)

Руководитель доцент ДПИиИИ, _____
(должность, уч. степень, уч. звание)
канд. физ.-мат. наук

_____ Ю.Е. Иванова
(подпись) (ФИО)

Практика пройдена в срок
с « 24 » июля 2023 г.
по « 05 » августа 2023 г.
на предприятии ФГБУН «Институт автоматики и процессов управления Дальневосточного отделения Российской академии наук»

Регистрационный № _____

_____
(подпись) (ФИО)
«____» _____202 г.

Защищен(а) с оценкой

« 05 » августа 2023 г.

г. Владивосток
2023

Оглавление

Введение

Учебно-технологическая (проектно-технологическая) практика проходила в ФГБУН «Институт автоматики и процессов управления Дальневосточного отделения Российской академии наук» во втором семестре 2022-2023 учебного года с 24.07.2023 г. по 05.08.2023 г. (в объеме 2 недели, 3 з.е., 108 часов).

Руководитель от ИАПУ ДВО РАН – Манцыбора Александр Анатольевич, научный сотрудник лаборатории № 52 ИАПУ ДВО РАН, кандидат физ.-мат. наук.

Целями учебно-технологической (проектно-технологической) практики являются: приобретение первичных практических умений и навыков по разработке проектов программных систем и проектной документации, а также знакомство с профессиональными задачами, решаемыми при создании программных систем.

Задачами учебно-технологической (проектно-технологической) практики являются:

– сбор и анализ требований заказчика к программному продукту;

– формализация предметной области программного проекта по результатам технического задания и экспресс-обследования;

– участие в проектировании компонентов программного продукта в объеме, достаточном для их конструирования в рамках поставленного задания;

– создание компонент программного обеспечения (кодирование, отладка, модульное и интеграционное тестирование);

– разработка тестового окружения, создание тестовых сценариев;

– разработка и оформление эскизной, технической и рабочей проектной документации.

Описание проекта программного средства

Проект программного средства представляет собой программную реализацию игры «Пятнашки».

Пятнашки – игра-головоломка, придуманная в 1878 году Ноем Чепмэном. Головоломка представляет собой набор из 15 одинаковых квадратных костяшек с нанесёнными на них числами, лежащих в квадратной коробке. Длина стороны коробки в четыре раза больше длины стороны костяшки, поэтому в коробке остаётся незаполненным одно квадратное поле. Цель игры – упорядочить костяшки по возрастанию номеров, перемещая их внутри коробки, желательно сделав как можно меньше перемещений.

Важно отметить, что перемещения можно совершать только за счет единственной пустой клетки на поле, на нее можно переместить только смежные с ней костяшки. Подразумевается, что человек не имеет возможности убрать все костяшки из коробки и расставить в нужном порядке или брать не смежную костяшку из коробки и ставить на пустое место.

Проект разработан с использованием принципов нисходящего проектирования; алгоритм программы основан на принципе структурного программирования.

В качестве среды разработки проекты была использована Microsoft Visual Studio. Языком реализации проекта является C++ с использованием свободной кроссплатформенной мультимедийной библиотеки SFML.

Работа выполнена командой из четырех человек: Дубровин А.А., Медушевская В.В., Поляков А.В., Тищенко Р.М.

Роли в команде распределены следующим образом:

№ п/п	ФИО	Роль
1	Дубровин А.А.	Реализатор, генератор идей, аналитик, координатор.
2	Медушевская В.В.	Завершитель, генератор идей, реализатор, аналитик.

3	Поляков А.В.	Аналитик, генератор идей, вдохновитель, координатор, завершитель.
4	Тищенко Р.М.	Реализатор, координатор, мотиватор, завершитель.

1. Неформальная постановка задачи

Реализовать программу, решающую следующую задачу:

Создать игру «Пятнашки» с визуальным оформлением в виде игрового поля и управляющего меню. Игровое поле представляет собой окно с клетками размерности 4x4, которые при запуске игры будут случайным образом заполняться квадратами с номерами от 1 до 15 включительно. Таким образом, лишь одна клетка на поле должна оставаться незаполненной.

Эта клетка нужна для осуществления перемещения кубиков. Пользователь сможет управлять перемещением кубиков при помощи кнопок клавиатуры «UP», «LEFT», «DOWN», «RIGHT».

Все взаимодействие с пользователем происходит через графическое окно игры, сообщения также выводятся на экран.

Меню представляет собой пункты: правила игры, начать игру, выйти из игры.

При нажатии клавиши «R» на экран выводятся правила игры, при этом игровой процесс приостанавливается. При нажатии кнопки «Back» правила игры скрываются, игровой процесс продолжается.

Перемещение может быть выполнено успешно, только если сдвигаемый кубик является смежным по отношению к свободной клетке. Смежность определяется таким образом, что свободная клетка должна соприкасаться одной из сторон с кубиком, который на ее место необходимо поставить. Иначе клетки являются не смежными.

На экране для пользователя в верхнем левом углу будет отображаться количество сделанных перемещений кубиков.

Цель игры: добиться расположения кубиков без пробелов в порядке от 1 до 15 таким образом, что номер «1» находился в первом ряду на крайнем левом месте, а номер «15» находился в четвертом ряду на третьем месте при отсчете слева. При этом пустая клетка оказывается в четвертом ряду на крайнем правом месте.

Такое расположение является условием победы и одновременно завершает игровой процесс. Далее пользователь может начать новую игру или выйти из игры через основное управляющее меню.

2. Формальная постановка задачи

Входные данные:

GraphicsWindow - графическое окно размерности 800x600.

StartMenuKeys = {«Start Game», «Rules», «Exit»} - кнопки начального игрового меню.

StartMenu - стартовое игровое меню.

StartMenuKeys ∈ StartMenu

ControlKeys = {«UP», «LEFT», «DOWN», «RIGHT», «R»} - управляющие клавиши клавиатуры.

MousePos = {posX, posY} - положение курсора мыши.

Выходные данные:

NewGraphicsWindow - перерисованное графическое окно.

WinMenuKeys = {«Restart Game», «Rules», «Exit»} - кнопки конечного игрового меню.

PlayGround - встроенное игровое поле.

PlayGround ∈ NewGraphicsWindow

Square = {$square_i$: $square_i$ = {$SquareColor_i$, $NumColor_i$, $SquareFrame_i$, $NumFrame_i$, $SqareNum_i$, $SquarePos_i = SquarePos_i(x_i, y_i)$, $x_i, y_i \in 0, \ldots, 3\}$, $i = 1, 2, \ldots, 16\}$ - множество квадратов с номерами от 1 до 16 ∈ PlayGround, где:

$SquareColor_i$ - цвет $square_i$;

$NumColor_i$ - цвет числа на $square_i$;

$SquareFrame_i$ - цвет рамки $square_i$;

$NumFrame_i$ - цвет обводки $SquareNum_i$;

$SquareNum_i \in N$, $SquareNum_i \in (1; 16)$ - номер $square_i$;

$SquarePos_i(x_i, y_i)$ - задает положение $square_i$ на игровом поле;

$square_{16}$ - полностью прозрачный. $SquareNum_{16} = 16$.

$SquareColor_i$ задаётся так, чтобы все цвета $square_i$ отличались. В процессе игры не меняются.

$MoveCounter \in (N + \{0\})$ - счетчик ходов.

RulesWindow = {«- The game consists of 15 tiles on a 4x4 board.
- The tiles have numbers from 1 to 15.
- You can move tiles using the arrow keys on your keyboard.
- To complete the game, you need to decompose all the tiles in ascending numbers on them.», «Back»} - окно с правилами игры, «Back» - возвращение в меню.

Формализованные связи:

Если координаты мыши posX, posY совпадают с кнопками Start-MenuKeys, WinMenuKeys, «Back», то кнопки изменяют цвет.

Если нажата кнопка «Start Game», то NewGraphicsWindow, PlayGround.

Если нажата кнопка «Rules», то RulesWindow.

Если нажата кнопка «Exit», то выход.

Если нажата клавиша «UP», то для k: $y_{16} = y_k - 1$ и $x_{16} = x_k$ выполняется: $Z = SqareNum_{16}$, $SqareNum_{16} = SqareNum_k$, $SqareNum_k = Z$, MoveCounter = MoveCounter + 1.

Если нажата клавиша «DOWN», то для k: $y_{16} = y_k + 1$ и $x_{16} = x_k$ выполняется:
$Z = SqareNum_{16}$, $SqareNum_{16} = SqareNum_k$, $SqareNum_k = Z$, MoveCounter = MoveCounter + 1.

Если нажата клавиша «LEFT», то для k: $y_{16} = y_k$ и $x_{16} = x_k - 1$ выполняется:
$Z = SqareNum_{16}$, $SqareNum_{16} = SqareNum_k$, $SqareNum_k = Z$, MoveCounter = MoveCounter + 1.

Если нажата клавиша «RIGHT», то для k: $y_{16} = y_k$ и $x_{16} = x_k + 1$ выполняется:

$Z = SqareNum_{16}$, $SqareNum_{16} = SqareNum_k$, $SqareNum_k = Z$, MoveCounter = MoveCounter + 1.

Если нажата клавиша «R», то RulesWindow.

Если нажата кнопка «Restart Game», то NewGraphicsWindow, PlayGround.

3. Алгоритм решения задачи

1. Начало.

2. Отрисовка окна игры с открытым на нем начальным управляющим меню.

3. Осуществление выбора из пунктов меню.

4. Если выбран пункт «Начать игру», то

   4.1.Скрыть меню.

   4.2.Переход на пункт 7.

5. Если выбран пункт «Правила игры», то

   5.1.Вывод пользователю на экран правил игры.

   5.2.Если нажата кнопка «Back», то

      5.2.1. Скрыть правила игры.

      5.2.2. Возврат в меню.

6. Если выбран пункт «Выход из игры», то

   6.1.Переход на КОНЕЦ алгоритма, пункт 11.

7. Случайная генерация положения кубиков на игровом поле.

8. Отрисовка графического окна с игровым полем и «разбросанными» кубиками.

9. Если нажата клавиша «R», то

   9.1.Отобразить правила игры.

   9.2.Если нажата кнопка «Back», то

      9.2.1. Скрыть правила.

9.3. Иначе выбор возможного перемещения кубика на свободную клетку.

9.4.Если нажата клавиша «UP», то

9.4.1. Кубик над пустой клеткой перемещается вниз.

9.5. Если нажата клавиша «LEFT», то

9.5.1. Кубик над пустой клеткой перемещается вправо.

9.6.Если нажата клавиша «DOWN», то

9.6.1. Кубик над пустой клеткой перемещается вверх.

9.7. Если нажата клавиша «RIGHT», то

9.7.1. Кубик над пустой клеткой перемещается влево.

10. Если положение кубиков не удовлетворяет условию победы (расположение от 1 до 15 по порядку) то

10.1. Переход на пункт 9.

10.2. Иначе, остановка игрового процесса.

10.3. Вывод сообщения пользователю на экран о том, что победа достигнута за N-ное количество шагов.

10.4. Отрисовка окна игры с открытым на нем конечным управляющим меню.

10.5. Осуществление выбора из пунктов меню.

10.6. Если выбран пункт “Начать игру заново”, то

10.6.1. Скрыть меню.

10.6.2. Переход на пункт 4.

10.7. Если выбран пункт «Правила игры», то

10.7.1. Вывод пользователю на экран правил игры.

10.7.2. Если нажата кнопка «Back», то

10.7.2.1.Скрыть правила игры.

10.7.2.2.Возврат в меню.

10.8. Если выбран пункт «Выход из игры», то

10.8.1. Переход на КОНЕЦ алгоритма, пункт 11.

11. Конец.

4. Спецификация данных

Входные данные:

Ввод данных осуществляется с клавиатуры и при помощи мыши.

С клавиатуры при помощи клавиш UP, DOWN, RIGHT, LEFT, R, где клавиши UP, DOWN, RIGHT, LEFT выполняют перемещение кубика, находящегося над пустым местом, вниз, вправо, влево, вверх соответственно, клавиша R вызывает окно с правилами игры.

MousePos{posX, posY} - управление игровым меню.

Выходные данные:

GraphicsWindow - графическое окно.

NewGraphicsWindow - перерисованное графическое окно.

Вывод всех следующих сообщений осуществляется на экран.

Сообщение 1: «- The game consists of 15 tiles on a 4x4 board.
- The tiles have numbers from 1 to 15.
- You can move tiles using the arrow keys on your keyboard.
- To complete the game, you need to decompose all the tiles in ascending numbers on them.».

Сообщение 2: «YOU WIN!!!».

Сообщение 3: «Total moves:» + MoveCounter.

Спецификация функций

1. Отрисовка окна с начальным игровым меню.
2. Осуществление выбора из пунктов меню.
3. Отрисовка окна с игровыми правилами.
4. Отрисовка окна с игровым полем.
5. Осуществление выхода из игры.
6. Случайная генерация положения кубиков на игровом поле.
7. Перемещение кубиков.
8. Подсчет ходов.
9. Отрисовка окна с конечным игровым меню.

6. Проект программного средства

Графическая иерархическая схема функциональных блоков программы представлена на рисунке 1.

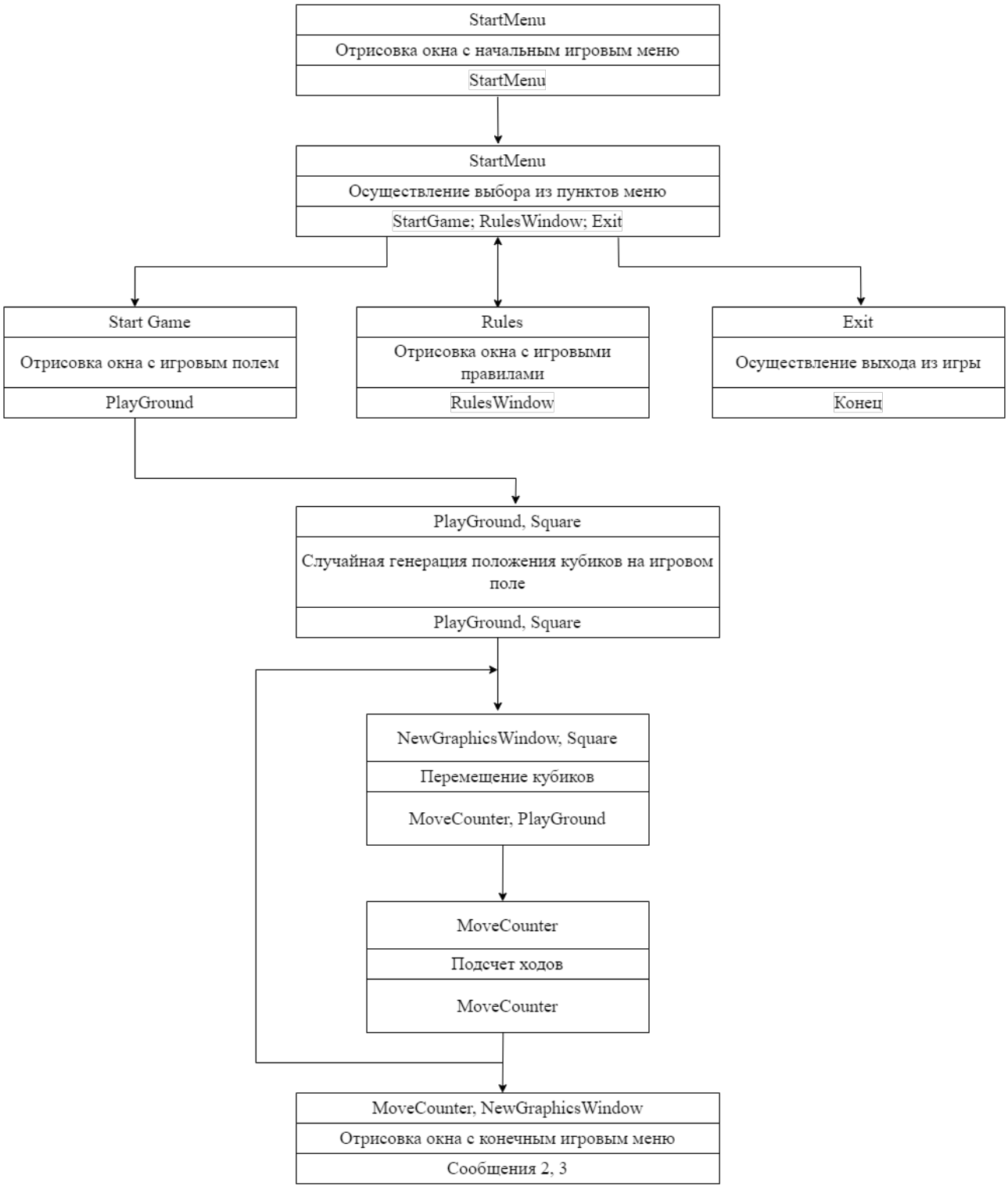


Рисунок 1 – Графическая иерархическая схема функциональных блоков программы

7. Описание данных программы

Идентификатор	Назначение	Описание
Класс Tile	Представляет объект квадрата в игре. Он унаследован от классов sf::RectangleShape и sf::Transformable из библиотеки SFML, что позволяет использовать его как объект для отрисовки и трансформации.	
Публичные методы класса Tile		
setTileSize	Задает размер прямоугольника фона квадрата и размер символов текста с числом.	void
Локальный контекст функции setTileSize		
rect	Параметр, представляющий размер (ширину и высоту) плитки. Этот параметр используется для установки размеров rectShape, который представляет визуальное отображение плитки на игровом поле.	Vector2f
text	Параметр, представляющий размер шрифта для числа на плитке. Этот параметр используется для установки размера шрифта numberText, который отображает число на плитке.	int
setTileFont	Устанавливает шрифт для текста с числом.	void
Локальный контекст функции setTileFont		
font	Параметр-ссылка на константный объект типа sf::Font. Этот параметр представляет	Font&

	шрифт, который будет использоваться для отображения числовой метки на плитке.	
setTileOrigin	Задает точку привязки для прямоугольника фона квадрата и текста с числом.	void
Локальный контекст функции setTileOrigin		
origin	Параметр, представляющий координаты точки происхождения (origin) для плитки.	Vector2f
setTileFillColor	Устанавливает цвет заполнения для прямоугольника фона квадрата и текста с числом.	void
Локальный контекст функции setTileFillColor		
rectCol	Параметр, представляющий цвет заливки для прямоугольника (rectShape) плитки.	Color
textCol	Параметр, представляющий цвет заливки для текста (numberText) на плитке.	Color
getTileRectFillColor	Возвращает цвет заполнения прямоугольника фона квадрата.	Color
getTileNumberFillColor	Возвращает цвет заполнения текста с числом.	Color
setTileOutlineThickness	Задает толщину контура для прямоугольника фона квадрата и текста с числом.	void
Локальный контекст функции setTileOutlineThickness		
rect	Представляет толщину обводки прямоугольника.	int
num	Представляет толщину	int

	обводки текста номера прямоугольника.	
setTileOutlineColor	Устанавливает цвет контура для прямоугольника фона квадрата и текста с числом.	void
getTileRectOutlineColor()	Возвращает цвет контура прямоугольника фона квадрата.	Color
Color getTileNumberOutlineColor()	Возвращает цвет контура текста с числом.	Color
setTilePosition	Задает позицию квадрата и обновляет позицию текста с числом соответственно. Метод также учитывает, является ли число на квадрате однозначным или двузначным для правильного выравнивания.	void
getTilePosition	Возвращает координаты квадрата.	Vector2f
setNum	Устанавливает значение числа, которое будет отображаться на квадрате.	void
getNum	Возвращает целочисленное значение, которое отображается на квадрате.	int
Приватные переменные класса Tile:		
rectShape	Объект прямоугольника SFML, представляющий фон квадрате.	RectangleShape
numberText	Объект текста SFML, представляющий число, отображаемое на квадрате.	Text
numberInt	Целочисленное значение числа, отображаемого на квадрате.	int
position	Координаты квадрата.	Vector2f

Приватный метод класса Tile:		
draw	Переопределяет метод draw из SFML для рендеринга объекта кнопки. Он отрисовывает как фоновый прямоугольник, так и текст на указанной целевой поверхности target с использованием предоставленных states.	Virtual void
Класс Button	Представляет собой кнопку в игре, реализованную с помощью библиотеки SFML. Унаследован от классов sf::RectangleShape и sf::Transformable, что позволяет использовать его как объект для отрисовки и трансформации.	
Конструкторы класса Button		
Button()	Пустой конструктор по умолчанию.	
Button(sf::Vector2f size, sf::Color btnCol)	Принимает размер кнопки (вектор размера) и цвет заполнения кнопки (btnCol). Создает кнопку с указанными параметрами.	Button
Button(sf::Vector2f size, sf::Color btnCol, std::string text, sf::Font& font, int charSize, sf::Color textCol)	Принимает размер кнопки (вектор размера), цвет заполнения кнопки (btnCol), текст, который будет отображаться на кнопке (text), шрифт (font), размер символов текста (charSize) и цвет текста (textCol). Создает кнопку с указанными параметрами и добавляет текст на кнопку.	Button
Публичные методы класса Button		

setRectColor	Устанавливает цвет заполнения кнопки.	void
setTextColor	Устанавливает цвет текста на кнопке.	void
setRectOutlineThickness	Задает толщину контура кнопки.	void
setTextOutlineThickness	Задает толщину контура текста на кнопке.	void
setRectOutlineColor	Устанавливает цвет контура кнопки.	void
setTextOutlineColor	Устанавливает цвет контура текста на кнопке.	void
setPosition	Задает позицию кнопки и обновляет позицию текста на кнопке, чтобы он оставался выровненным по центру кнопки.	void
isMouseOver	Проверяет, находится ли курсор мыши над кнопкой в данный момент. Возвращает true, если курсор находится над кнопкой, и false в противном случае.	bool
Приватные переменные класса Button		
btnRect	Объект прямоугольника SFML, представляющий фон кнопки.	RectangleShape
btnText	Объект текста SFML, представляющий текст на кнопке.	Text
Приватный метод класса Button		
draw	Переопределяет метод draw из SFML для рендеринга объекта кнопки. Он отрисовывает как фоновый прямоугольник, так и текст на указанной це-	virtual function

	левой поверхности target с использованием предоставленных states.	
Функция main		
Переменные функции main		
gameWidth	Константа, определяющая ширину игрового окна.	float
gameHeight	Константа, определяющая высоту игрового окна.	float
tagWidth	Ширина плиток.	float
tagHeight	Высота плиток.	float
menuLayer	Флаг, указывающий на текущий слой меню.	bool
winLayer	Флаг, указывающий на текущий слой победы.	bool
playLayer	Флаг, указывающий на текущий слой игры.	bool
rulesLayer	Флаг, указывающий на текущий слой правил.	bool
emptyX	Переменная, хранящая позицию X пустой плитки.	int
emptyY	Переменная, хранящая позицию Y пустой плитки.	int
helpTile	Объект класса Tile, используется для временного хранения свойств.	Tile
moveCounter	Счетчик количества ходов игрока. Соответствует “MoveCounter” в формальной постановке.	int
window	Основное окно приложения. Соответствует “GraphicsWindow” в формальной постановке.	RenderWindow

fnt	Объект класса sf::Font для загрузки шрифта.	Font
fontDir	Путь к файлу шрифта.	string
bgRect	Объект класса sf::RectangleShape, используется для отображения фона игрового слоя.	RectangleShape
moveCounterText	Объект класса sf::Text, используется для отображения счетчика ходов.	Text
hintText	Объект класса sf::Text, используется для отображения подсказки.	Text
tile[4][4]	Массив 4x4 объектов класса Tile, представляющих плитки на игровом поле. Соответствует square[i] из множества Square в формальной постановке.	Tile
menuMsg	Объект класса sf::Text, используется для отображения текста на главном меню.	Text
startBtn	Объект класса Button, представляющий кнопку "Start Game" на главном меню. Соответствует элементу “Start Game” из множества “StartMenuKeys” в формальной постановке.	Button
rulesBtn	Объект класса Button, представляющий кнопку "Rules" на главном меню. Соответствует элементу “Rules” из множества	Button

	“StartMenuKeys” и “WinMenuKeys” в формальной постановке.	
exitBtn	Объект класса Button, представляющий кнопку "Exit" на главном меню. Соответствует элементу “Exit” из множества “StartMenuKeys” и “WinMenuKeys” в формальной постановке.	Button
rulesMsg	Объект класса sf::Text, используется для отображения текста с правилами.	Text
restartBtn	Объект класса Button, представляющий кнопку "Restart Game" на слое с правилами. Соответствует элементу “Restart Game” из множества “WinMenuKeys” в формальной постановке.	Button
winMsg	Объект класса sf::Text, используется для отображения сообщения о победе.	Text
backBtn	Объект класса Button, представляющий кнопку "Back" на слое с сообщением о победе. Соответствует “Back” в формальной постановке.	Button
Методы функции main:		
swapParameters	Определенная пользователем функция для	void

	обмена свойствами между двумя объектами типа Tile. Это используется для обмена визуальным представлением плиток при перемещении их игроком.	

8. Алгоритм программы на языке PDL

```
#include <SFML/Graphics.hpp>
#include <cmath>
#include <ctime>
#include <cstdlib>
#include <iostream>
#include <Windows.h>

class Tile : public sf::RectangleShape, public sf::Transformable
{
public:
   void setTileSize(sf::Vector2f rect, int text)
   {
      rectShape.setSize(rect);
      numberText.setCharacterSize(text);
   }
   void setTileFont(const sf::Font& font)
   {
      numberText.setFont(font);
   }
   void setTileOrigin(sf::Vector2f origin)
   {
      rectShape.setOrigin(origin);
      numberText.setOrigin(origin);
   }
   void setTileFillColor(sf::Color rectCol, sf::Color textCol)
   {
      rectShape.setFillColor(rectCol);
      numberText.setFillColor(textCol);
   }
   sf::Color getTileRectFillColor() { return rectShape.getFillColor(); }
   sf::Color getTileNumberFillColor() { return numberText.getFillColor(); }

   void setTileOutlineThickness(int rect, int num)
   {
      rectShape.setOutlineThickness(rect);
      numberText.setOutlineThickness(num);
   }
```

```
        void setTileOutlineColor(sf::Color rect, sf::Color num)
        {
            rectShape.setOutlineColor(rect);
            numberText.setOutlineColor(num);
        }
        sf::Color getTileRectOutlineColor() { return rectShape.getOutlineColor(); }
        sf::Color getTileNumberOutlineColor() { return numberText.getOutlineColor(); }
        void setTilePosition(sf::Vector2f pos)
        {
            rectShape.setPosition(pos);
            if (numberInt < 10) numberText.setPosition(sf::Vector2f(pos.x + 23, pos.y));
            else numberText.setPosition(sf::Vector2f(pos.x + 3, pos.y));
            position = pos;
        }
        sf::Vector2f getTilePosition() { return position; }
        void setNum(int num)
        {
            numberText.setString(std::to_string(num));
            numberInt = num;
        }
        int getNum() { return numberInt; }
    private:
        sf::RectangleShape rectShape;
        sf::Text numberText;
        int numberInt = 0;
        sf::Vector2f position;
        virtual void draw(sf::RenderTarget& target, sf::RenderStates states) const
        {
            target.draw(rectShape, states);
            target.draw(numberText, states);
        }
    };
    class Button : public sf::RectangleShape, public sf::Transformable
    {
    public:
        Button() { }
        Button(sf::Vector2f size, sf::Color btnCol)
        {
            btnRect.setSize(size);
            btnRect.setFillColor(btnCol);
        }
        Button(sf::Vector2f size, sf::Color btnCol, std::string text, sf::Font& font, int charSize,
sf::Color textCol)
        {
            btnRect.setSize(size);
            btnRect.setFillColor(btnCol);

            btnText.setString(text);
            btnText.setFont(font);
            btnText.setCharacterSize(charSize);
            btnText.setFillColor(textCol);
        }
```

```
        void setRectColor(sf::Color color)
        {
            btnRect.setFillColor(color);
        }
        void setTextColor(sf::Color color)
        {
            btnText.setFillColor(color);
        }

        void setRectOutlineThickness(float thickness)
        {
            btnRect.setOutlineThickness(thickness);
        }
        void setTextOutlineThickness(float thickness)
        {
            btnText.setOutlineThickness(thickness);
        }
        void setRectOutlineColor(sf::Color color)
        {
            btnRect.setOutlineColor(color);
        }
        void setTextOutlineColor(sf::Color color)
        {
            btnText.setOutlineColor(color);
        }
        void setPosition(sf::Vector2f pos)
        {
            btnRect.setPosition(pos);
            float textPosX = (pos.x + btnRect.getGlobalBounds().width / 2) - (btn-
Text.getGlobalBounds().width / 2);
            float textPosY = (pos.y + btnRect.getGlobalBounds().height / 2) - (btn-
Text.getGlobalBounds().height) + 4;
            btnText.setPosition(textPosX, textPosY);
        }
        bool isMouseOver(sf::RenderWindow& window)
        {
            float mouseX = sf::Mouse::getPosition(window).x;
            float mouseY = sf::Mouse::getPosition(window).y;
            float btnPosX = btnRect.getPosition().x;
            float btnPosY = btnRect.getPosition().y;
            float btnPosXWidth = btnRect.getPosition().x + btnRect.getLocalBounds().width;
            float btnPosYHeight = btnRect.getPosition().y + btnRect.getLocalBounds().height;
            if (mouseX > btnPosX && mouseX < btnPosXWidth && mouseY > btnPosY &&
mouseY < btnPosYHeight)
            {
                return true;
            }
            return false;
        }
    private:
        sf::RectangleShape btnRect;
        sf::Text btnText;
```

```
        virtual void draw(sf::RenderTarget& target, sf::RenderStates states) const
        {
            target.draw(btnRect, states);
            target.draw(btnText, states);
        }
    };
    void swapParameters(Tile& tile1, Tile& tile2)
    {
        Tile helpTile;
        helpTile.setTileFillColor(tile1.getTileRectFillColor(), tile1.getTileNumberFillColor());
        helpTile.setTileOutlineColor(tile1.getTileRectOutlineColor(),
tile1.getTileNumberOutlineColor());
        helpTile.setNum(tile1.getNum());
        tile1.setTileFillColor(tile2.getTileRectFillColor(), tile2.getTileNumberFillColor());
        tile1.setTileOutlineColor(tile2.getTileRectOutlineColor(),
tile2.getTileNumberOutlineColor());
        tile1.setNum(tile2.getNum());
        tile2.setTileFillColor(helpTile.getTileRectFillColor(), help-
Tile.getTileNumberFillColor());
        tile2.setTileOutlineColor(helpTile.getTileRectOutlineColor(), help-
Tile.getTileNumberOutlineColor());
        tile2.setNum(helpTile.getNum());
        tile1.setTilePosition(tile1.getTilePosition());
        tile2.setTilePosition(tile2.getTilePosition());
    }
    const float gameWidth = 800, gameHeight = 600;
    int main()
    {
        const float tagWidth = 80, tagHeight = 80;
        bool menuLayer = true;
        bool winLayer = false;
        bool playLayer = false;
        bool rulesLayer = false;
        int emptyX = 3; int emptyY = 3;
        Tile helpTile;
        int moveCounter = 0;
        sf::RenderWindow window(sf::VideoMode(gameWidth, gameHeight), "The 15 Puzzle
Game");
        window.setVerticalSyncEnabled(true);
        sf::Font fnt;
        std::string fontDir = "resources/tuffy.ttf";
        if (!fnt.loadFromFile(fontDir));

        sf::RectangleShape bgRect;
        bgRect.setSize(sf::Vector2f(400, 400));
        bgRect.setOrigin(200, 200);
        bgRect.setPosition(sf::Vector2f(gameWidth / 2, gameHeight / 2));
        bgRect.setFillColor(sf::Color(64, 64, 64));
        bgRect.setOutlineThickness(5);
        bgRect.setOutlineColor(sf::Color::Black);

        sf::Text moveCounterText;
```

```
        moveCounterText.setCharacterSize(50);
        moveCounterText.setFont(fnt);
        moveCounterText.setOutlineColor({ 0, 0, 0 });
        moveCounterText.setOutlineThickness(4);

        sf::Text hintText;
        hintText.setCharacterSize(45);
        hintText.setFont(fnt);
        hintText.setString("Press 'R' to open rules.");
        Tile tile[4][4];
        for (int i = 0; i < 4; i++)
        {
           for (int j = 0; j < 4; j++)
           {
              tile[j][i].setTileFont(fnt);
              tile[j][i].setNum((j + i * 4) + 1);
              tile[j][i].setTileSize(sf::Vector2f(tagWidth, tagHeight), 60);
              tile[j][i].setTileOrigin(sf::Vector2f(tagWidth / 2, tagHeight / 2));
              tile[j][i].setTileFillColor(sf::Color(255, 255 - (i * 4 + j) * 15, 255 - (i * 4 + j) *
15), sf::Color(0, 0, 0));
              tile[j][i].setTileOutlineThickness(5, 0);
              tile[j][i].setTileOutlineColor(sf::Color::Black, sf::Color::Transparent);
              tile[j][i].setTilePosition(sf::Vector2f((gameWidth / 2) - 150 + j * 100, (game-
Height / 2) - 150 + i * 100));
           }
        }
        tile[3][3].setTileFillColor(sf::Color::Transparent, sf::Color::Transparent);
        tile[3][3].setTileOutlineColor(sf::Color::Transparent, sf::Color::Transparent);
        sf::Text menuMsg;
        menuMsg.setFont(fnt);
        menuMsg.setString("The 15 puzzle game");
        menuMsg.setOutlineColor(sf::Color::Black);
        menuMsg.setOutlineThickness(5);
        menuMsg.setCharacterSize(64);
        menuMsg.setOrigin(sf::Vector2f(220, 50));
        menuMsg.setPosition(sf::Vector2f(gameWidth / 2 - 50, gameHeight / 4));

        Button startBtn(sf::Vector2f(250, 50), sf::Color::White, "Start Game", fnt, 38,
sf::Color::Black);
        startBtn.setPosition(sf::Vector2f(gameWidth / 2 - 125, gameHeight / 3 + 50));
        startBtn.setRectOutlineColor(sf::Color::Black);
        startBtn.setRectOutlineThickness(5);

        Button rulesBtn(sf::Vector2f(250, 50), sf::Color::White, "Rules", fnt, 38,
sf::Color::Black);
        rulesBtn.setPosition(sf::Vector2f(gameWidth / 2 - 125, gameHeight / 3 + 125));
        rulesBtn.setRectOutlineColor(sf::Color::Black);
        rulesBtn.setRectOutlineThickness(5);

        Button exitBtn(sf::Vector2f(250, 50), sf::Color::White, "Exit", fnt, 38,
sf::Color::Black);
        exitBtn.setPosition(sf::Vector2f(gameWidth / 2 - 125, gameHeight / 3 + 200));
```

```
        exitBtn.setRectOutlineColor(sf::Color::Black);
        exitBtn.setRectOutlineThickness(5);
        sf::Text rulesMsg;
        rulesMsg.setFont(fnt);
        rulesMsg.setString("- The game consists of 15 tiles on a 4x4 board.\n- The tiles have numbers from 1 to 15\n- You can move tiles using the arrow keys on your\n keyboard.\n- To complete the game, you need to decompose all \nthe tiles in ascending numbers on them.");
        rulesMsg.setCharacterSize(32);
        rulesMsg.setFillColor(sf::Color::White);
        rulesMsg.setPosition(sf::Vector2f(25, gameHeight / 5));

        Button restartBtn(sf::Vector2f(250, 50), sf::Color::White, "Restart Game", fnt, 38, sf::Color::Black);
        restartBtn.setPosition(sf::Vector2f(gameWidth / 2 - 125, gameHeight / 3 + 50));
        restartBtn.setRectOutlineColor(sf::Color::Black);
        restartBtn.setRectOutlineThickness(5);
        sf::Text winMsg;
        winMsg.setFont(fnt);
        winMsg.setPosition(sf::Vector2f(180, 100));
        winMsg.setCharacterSize(100);
        winMsg.setFillColor(sf::Color::White);
        winMsg.setOutlineColor(sf::Color::Black);
        winMsg.setOutlineThickness(5);
        winMsg.setString("YOU WIN!!!");

        Button backBtn(sf::Vector2f(150, 50), sf::Color::White, "Back", fnt, 38, sf::Color::Black);
        backBtn.setPosition(sf::Vector2f(25, 25));
        backBtn.setRectOutlineColor(sf::Color::Black);
        backBtn.setRectOutlineThickness(5);

        while (window.isOpen())
        {
            sf::Event event;
            while (window.pollEvent(event))
            {
                if (event.type == sf::Event::Closed)
                    window.close();

                if (event.type == sf::Event::MouseMoved)
                {
                    if (startBtn.isMouseOver(window)) startBtn.setRectColor({ 175, 175, 175 });
                    else startBtn.setRectColor(sf::Color::White);

                    if (rulesBtn.isMouseOver(window)) rulesBtn.setRectColor({ 175, 175, 175 });
                    else rulesBtn.setRectColor(sf::Color::White);

                    if (exitBtn.isMouseOver(window)) exitBtn.setRectColor({ 175, 175, 175 });
                    else exitBtn.setRectColor(sf::Color::White);

                    if (backBtn.isMouseOver(window)) backBtn.setRectColor({ 175, 175, 175 });
                    else backBtn.setRectColor(sf::Color::White);
```

```
                if (restartBtn.isMouseOver(window)) restartBtn.setRectColor({ 175, 175, 175 });
                else restartBtn.setRectColor(sf::Color::White);
            }
            if (event.type == sf::Event::MouseButtonPressed)
            {
                if ((startBtn.isMouseOver(window) || restartBtn.isMouseOver(window)) && (menuLayer || winLayer))
                {
                    srand(time(0));
                    int direction;
                    menuLayer = false; winLayer = false;
                    moveCounter = 0;
                    for (int i = 0; i < 1000; i++)
                    {
                        direction = rand() % 4;
                        switch (direction)
                        {
                        case 0:
                        {
                            if (emptyX > 0)
                            {
                                swapParameters(tile[emptyX][emptyY], tile[emptyX - 1][emptyY]);
                                emptyX = emptyX - 1;
                                break;
                            }
                        }
                        case 1:
                        {
                            if (emptyX < 3)
                            {
                                swapParameters(tile[emptyX][emptyY], tile[emptyX + 1][emptyY]);
                                emptyX = emptyX + 1;
                                break;
                            }
                        }
                        case 2:
                        {
                            if (emptyY < 3)
                            {
                                swapParameters(tile[emptyX][emptyY], tile[emptyX][emptyY + 1]);
                                emptyY = emptyY + 1;
                                break;
                            }
                        }
                        case 3:
                        {
                            if (emptyY > 0)
                            {
                                swapParameters(tile[emptyX][emptyY], tile[emptyX][emptyY - 1]);
                                emptyY = emptyY - 1;
```

```
                        break;
                    }
                }

                default:
                    break;
                }
            }
            while (emptyX < 3)
            {
                swapParameters(tile[emptyX][emptyY], tile[emptyX + 1][emptyY]);
                emptyX = emptyX + 1;
            }
            while (emptyY < 3)
            {
                swapParameters(tile[emptyX][emptyY], tile[emptyX][emptyY + 1]);
                emptyY = emptyY + 1;
            }
            playLayer = true;
        }
        if (rulesBtn.isMouseOver(window) && (menuLayer || winLayer)) rulesLayer =
true;
        if (exitBtn.isMouseOver(window) && (menuLayer || winLayer)) win-
dow.close();

        if (backBtn.isMouseOver(window) && rulesLayer) rulesLayer = false;
    }

    if (playLayer && !winLayer)
    {
        //Controls (left, right, down, up, rules)
        if ((event.type == sf::Event::KeyPressed) &&
            (event.key.code == sf::Keyboard::Left) &&
            emptyX > 0 && (playLayer && !rulesLayer)) //Left
        {
            swapParameters(tile[emptyX][emptyY], tile[emptyX - 1][emptyY]);
            moveCounter++;
            emptyX = emptyX - 1;
        }

        if ((event.type == sf::Event::KeyPressed) &&
            (event.key.code == sf::Keyboard::Right) &&
            emptyX < 3 && (playLayer && !rulesLayer)) //Right
        {
            swapParameters(tile[emptyX][emptyY], tile[emptyX + 1][emptyY]);
            moveCounter++;
            emptyX = emptyX + 1;

            winLayer = true;
            playLayer = false;
            for (int i = 0; i < 4; i++)
            {
```

```
                for (int j = 0; j < 4; j++)
                {
                    if (tile[j][i].getNum() != (j + i * 4) + 1)
                    {
                        winLayer = false;
                        playLayer = true;
                    }
                }
            }
        }

        if ((event.type == sf::Event::KeyPressed) &&
            (event.key.code == sf::Keyboard::Down) &&
            emptyY < 3 && (playLayer && !rulesLayer)) //Down
        {
            swapParameters(tile[emptyX][emptyY], tile[emptyX][emptyY + 1]);
            moveCounter++;
            emptyY = emptyY + 1;

            winLayer = true;
            playLayer = false;
            for (int i = 0; i < 4; i++)
            {
                for (int j = 0; j < 4; j++)
                {
                    if (tile[j][i].getNum() != (j + i * 4) + 1)
                    {
                        winLayer = false;
                        playLayer = true;
                    }
                }
            }
        }

        if ((event.type == sf::Event::KeyPressed) &&
            (event.key.code == sf::Keyboard::Up) &&
            emptyY > 0 && (playLayer && !rulesLayer)) //Up
        {
            swapParameters(tile[emptyX][emptyY], tile[emptyX][emptyY - 1]);
            moveCounter++;
            emptyY = emptyY - 1;
        }

        if ((event.type == sf::Event::KeyPressed) &&
            (event.key.code == sf::Keyboard::R) &&
            (playLayer && !rulesLayer)) //Rules
        {
            rulesLayer = true;
        }
    }
}
```

```
        window.clear(sf::Color(128, 128, 128));
        if (rulesLayer) //Rules layer draw
        {
            window.draw(rulesMsg);
            window.draw(backBtn);
        }
        else if (menuLayer) //Main menu draw
        {
            window.draw(menuMsg);
            window.draw(startBtn);
            window.draw(rulesBtn);
            window.draw(exitBtn);
        }
        else if (winLayer) //Win layer draw
        {
            window.draw(winMsg);
            window.draw(restartBtn);
            window.draw(rulesBtn);
            window.draw(exitBtn);

            moveCounterText.setString("Total moves: " + std::to_string(moveCounter));
            sf::FloatRect textRect = moveCounterText.getGlobalBounds();
            moveCounterText.setOrigin(textRect.width / 2, textRect.height / 2);
            moveCounterText.setPosition(sf::Vector2f(gameWidth / 2, gameHeight - 65));
            window.draw(moveCounterText);
        }
        else if (playLayer) //Play layer draw
        {
            window.draw(bgRect);
            for (int i = 0; i < 4; i++)
            {
                for (int j = 0; j < 4; j++)
                {
                    window.draw(tile[j][i]);
                }
            }
            moveCounterText.setString("Moves: " + std::to_string(moveCounter));
            sf::FloatRect moveTextRect = moveCounterText.getGlobalBounds();
            moveCounterText.setOrigin(moveTextRect.width / 2, moveTextRect.height / 2);
            moveCounterText.setPosition(sf::Vector2f(gameWidth / 2, gameHeight - 65));
            window.draw(moveCounterText);
            sf::FloatRect hintTextRect = hintText.getGlobalBounds();
            hintText.setOrigin(hintTextRect.width / 2, hintTextRect.height / 2);
            hintText.setPosition(sf::Vector2f(gameWidth / 2, 50));
            window.draw(hintText);
        }
        window.display();
    }
    return 0;
}
```

9. Тесты

Методом тестирования является черный ящик.

1. Нажатие левой кнопки мыши в произвольной области экрана, не являющейся кнопкой. Действие происходит в меню.

Результат: ничего не произошло.

2. Нажатие клавиши клавиатуры в меню.

Результат: ничего не произошло.

3. Нажатие на кнопку «rules».

Результат: переход в раздел «правила», вывод сообщения и кнопок. Программа работает корректно.

4. Нажатие левой кнопки мыши в произвольной области экрана, не являющейся кнопкой. Действие происходит в разделе «правила».

Результат: ничего не произошло.

5. Нажатие клавиши клавиатуры в разделе «правила».

Результат: ничего не произошло.

6. Нажатие на кнопку «back» в разделе «правила».

Результат: переход в раздел «меню». Программа работает корректно.

7. Нажатие на кнопку «Exit» в меню.

Результат: Завершение работы программы. Программа работает корректно.

8. Нажатие на кнопку «Start game» в меню.

Результат: переход в раздел "Игра", вывод игрового поля и вывод сообщения. Программа работает корректно.

9. Нажатие любой кнопки мыши в произвольной области экрана, находясь в разделе "Игра".

Результат: ничего не произошло.

10. Нажатие клавиши клавиатуры, не являющейся клавишей управления, находясь в разделе «Игра».

Результат: ничего не произошло.

11. Нажатие клавиши «R» в разделе «Игра».

Результат: переход в раздел «правила», вывод сообщения и кнопок. Программа работает корректно.

12. Нажатие клавиши «DOWN» в разделе «Игра».

Результат: ничего не произошло. Программа работает корректно. Недопустимое движение.

13. Нажатие клавиши «RIGHT» в разделе «Игра».

Результат: ничего не произошло. Программа работает корректно. Недопустимое движение.

14. Нажатие клавиши «стрелка вверх» в разделе «Игра».

Результат: пустой квадрат передвинулся вверх на одну клетку, счетчик поменялся. Программа работает корректно.

15. Нажатие клавиши «стрелка влево» в разделе «Игра».

Результат: пустой квадрат передвинулся влево на одну клетку, счетчик поменялся. Программа работает корректно.

16. Нажатие клавиши «стрелка вверх» в разделе «Игра».

Результат: ничего не произошло. Программа работает корректно. Недопустимое движение.

17. Нажатие клавиши «стрелка влево» в разделе «Игра».

Результат: ничего не произошло. Программа работает корректно. Недопустимое движение.

18. Нажатие клавиши «стрелка вниз» в разделе «Игра».

Результат: пустой квадрат передвинулся вниз на одну клетку, счетчик поменялся. Программа работает корректно.

19. Нажатие клавиши «стрелка вправо» в разделе «Игра».

Результат: пустой квадрат передвинулся вправо на одну клетку, счетчик поменялся. Программа работает корректно.

20. Нажатие клавиши «стрелка вправо» в разделе «Игра» при данном расположении.

Результат: пустой квадрат передвинулся вправо на одну клетку, счетчик поменялся, переход в раздел «конец игры», вывод счетчика и сообщения. Программа работает корректно.

21. Нажатие левой кнопки мыши в произвольной области экрана, не являющейся кнопкой. Действие происходит в разделе «конец игры».

Результат: ничего не произошло.

22. Нажатие клавиши клавиатуры в разделе «конец игры».

Результат: ничего не произошло.

23. Нажатие на кнопку «Restart Game» в разделе «конец игры».

Результат: переход в раздел «Игра», вывод игрового поля и вывод сообщения. Программа работает корректно.

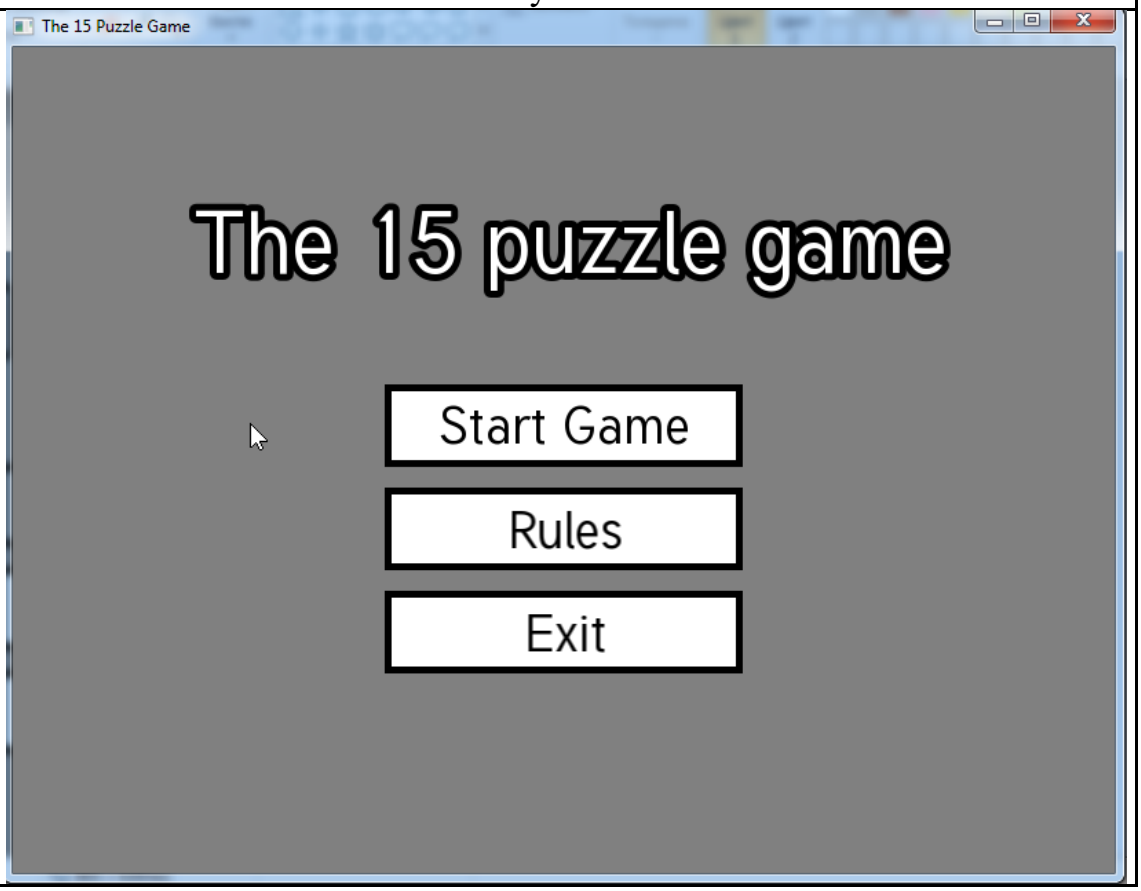
24. Нажатие на кнопку «rules» в разделе «конец игры».

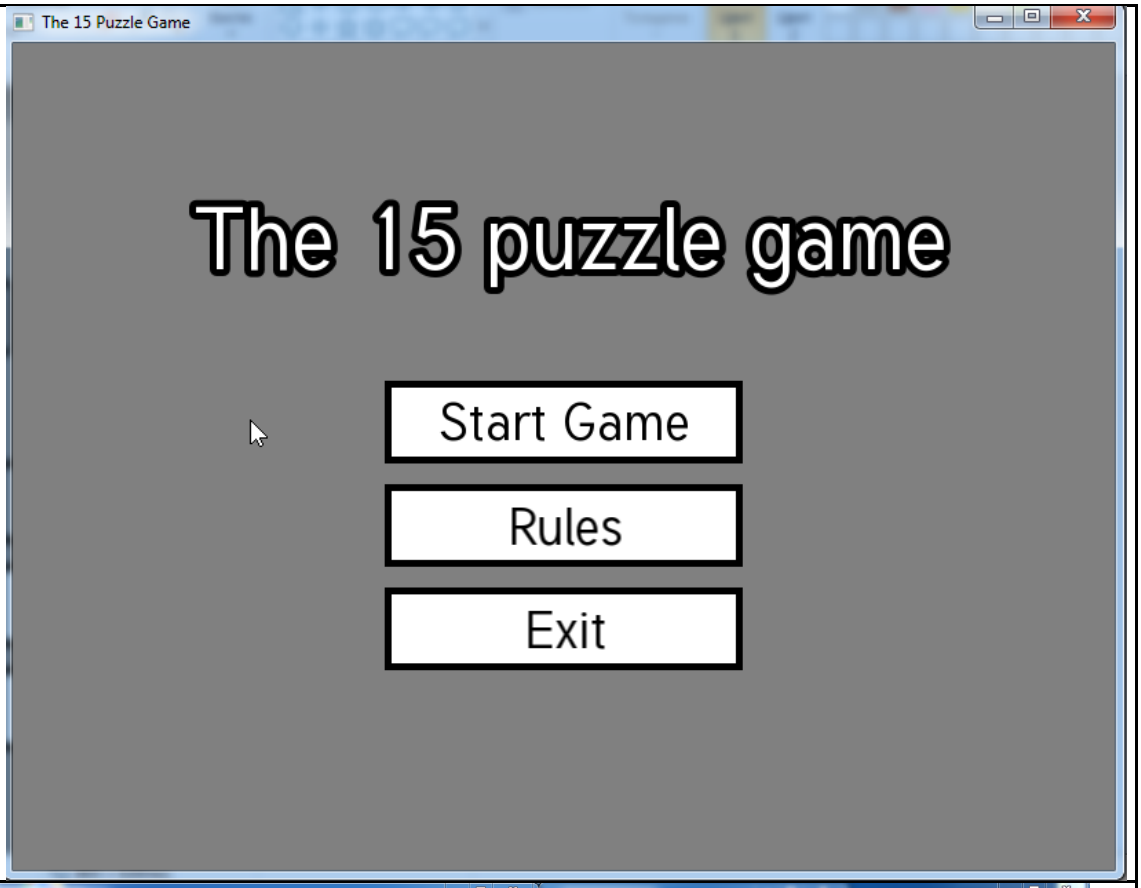
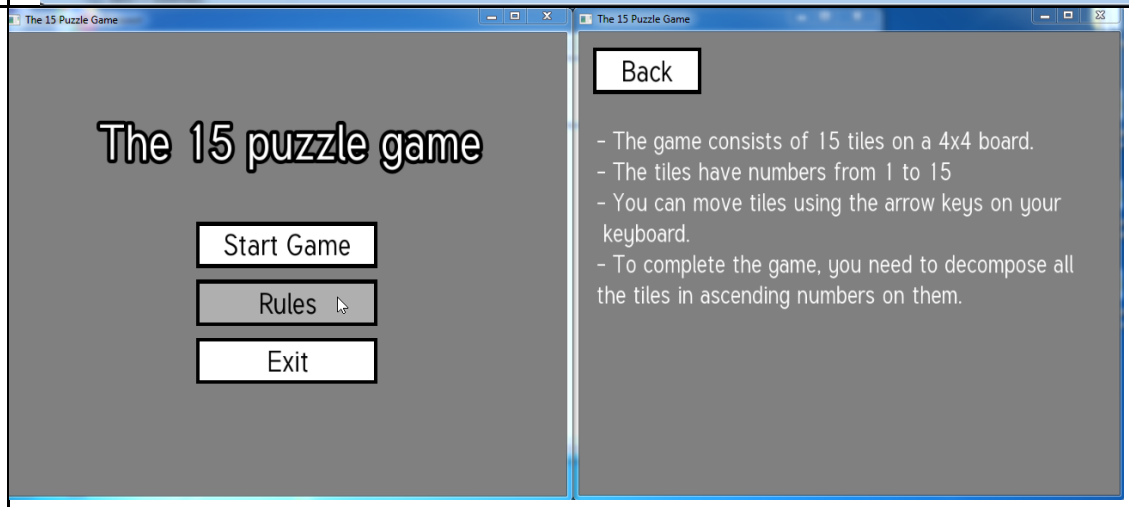
Результат: переход в раздел «правила», вывод сообщения и кнопок. Программа работает корректно.

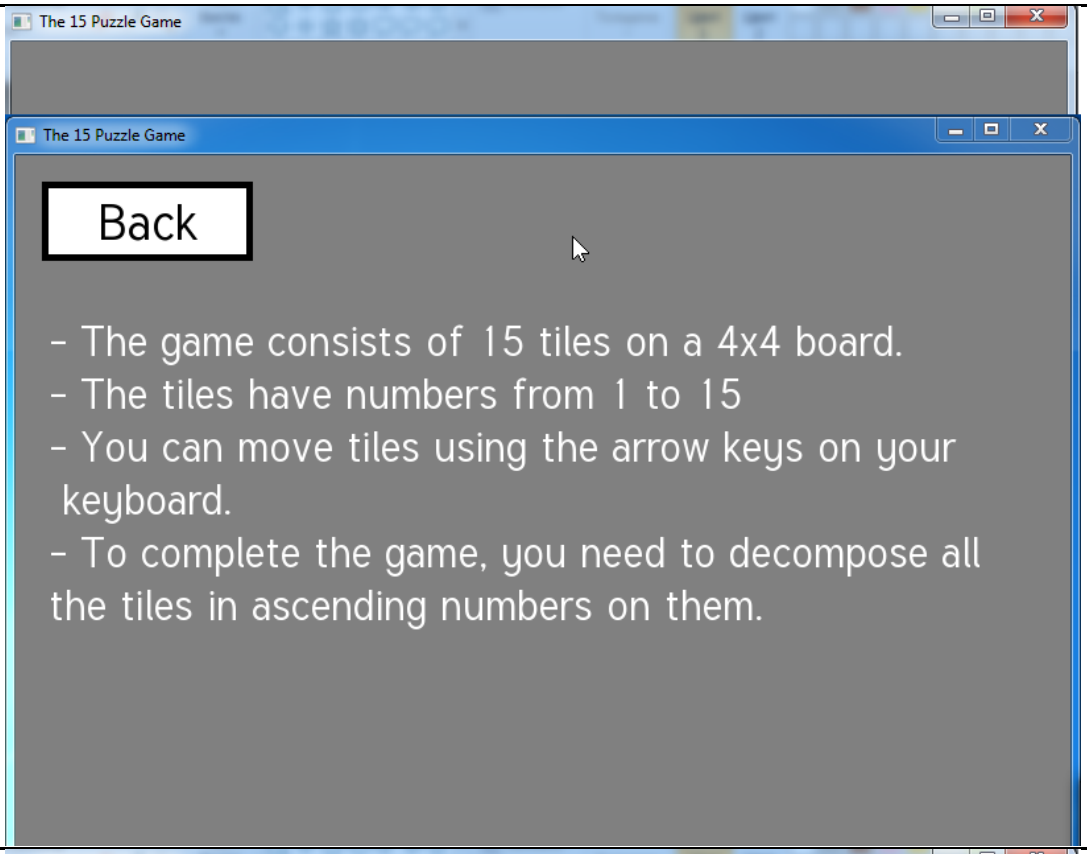
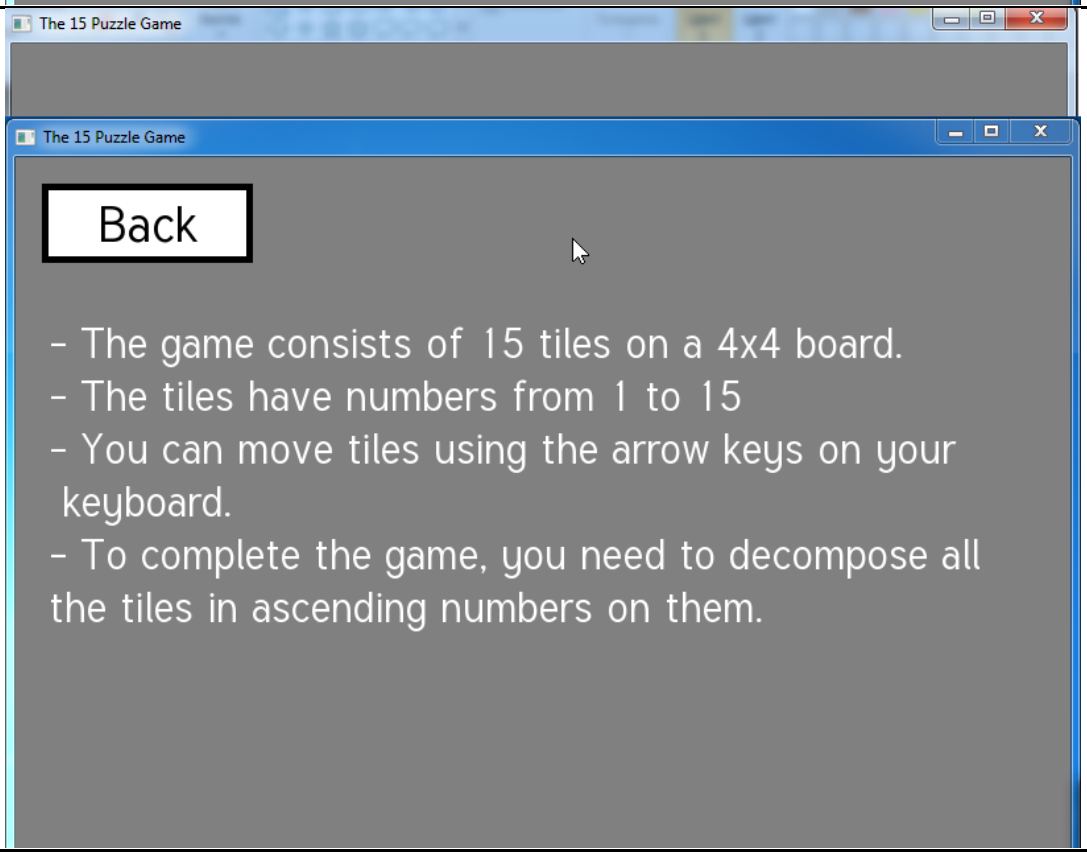
25. Нажатие на кнопку «Exit» в разделе «конец игры».

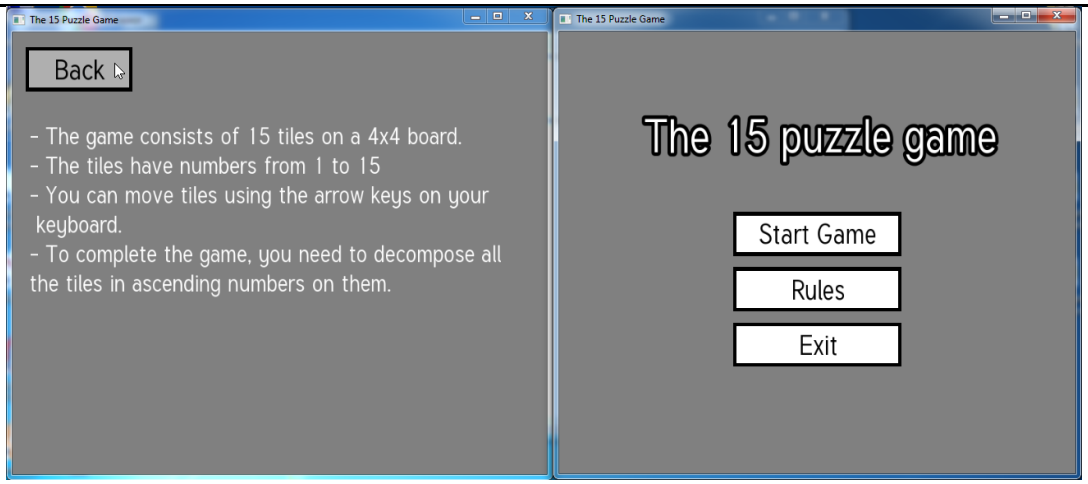
Результат: Завершение работы программы. Программа работает корректно.

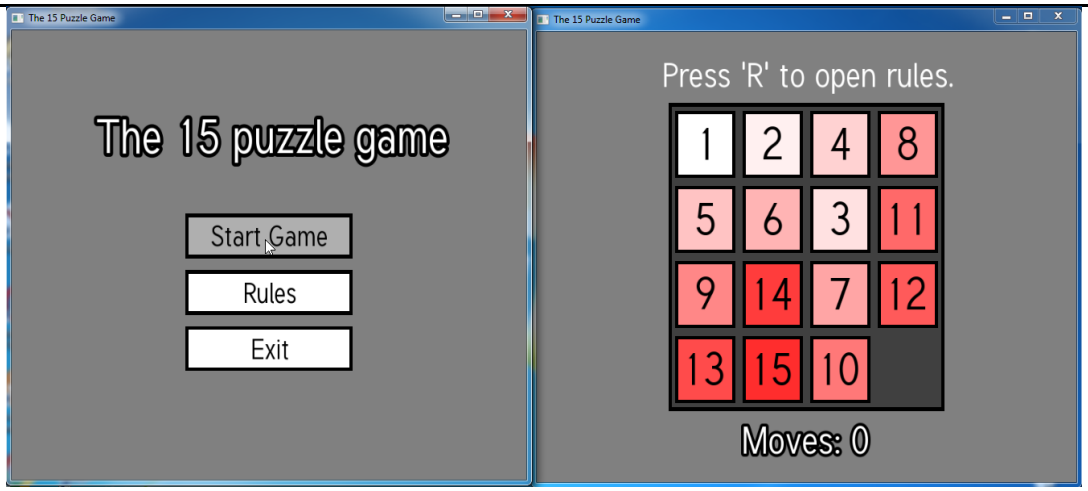
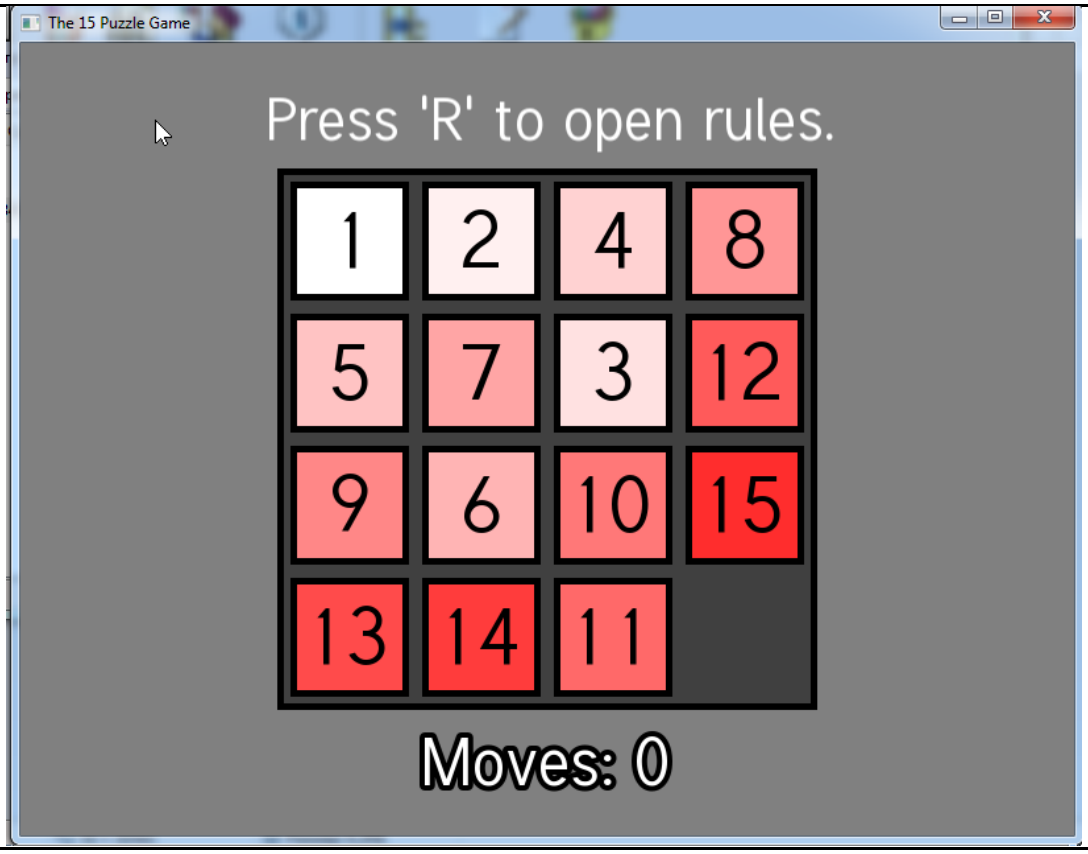
10. Тестирование программы

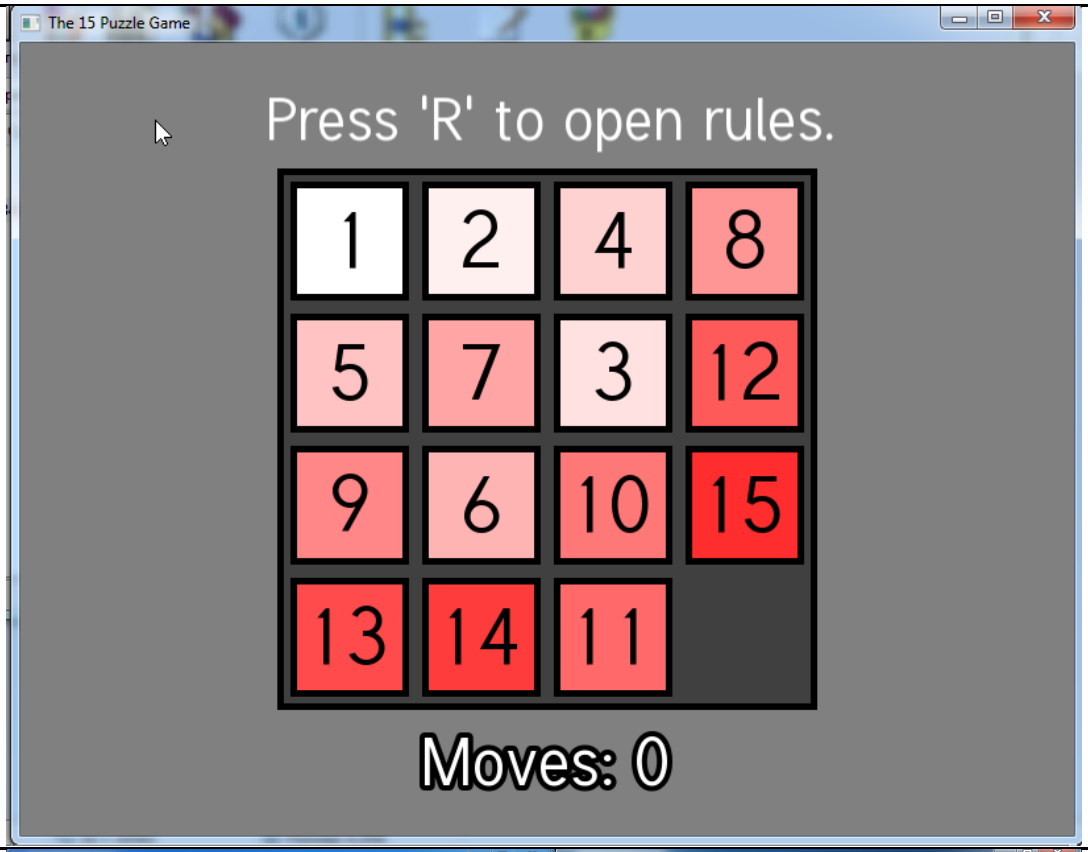
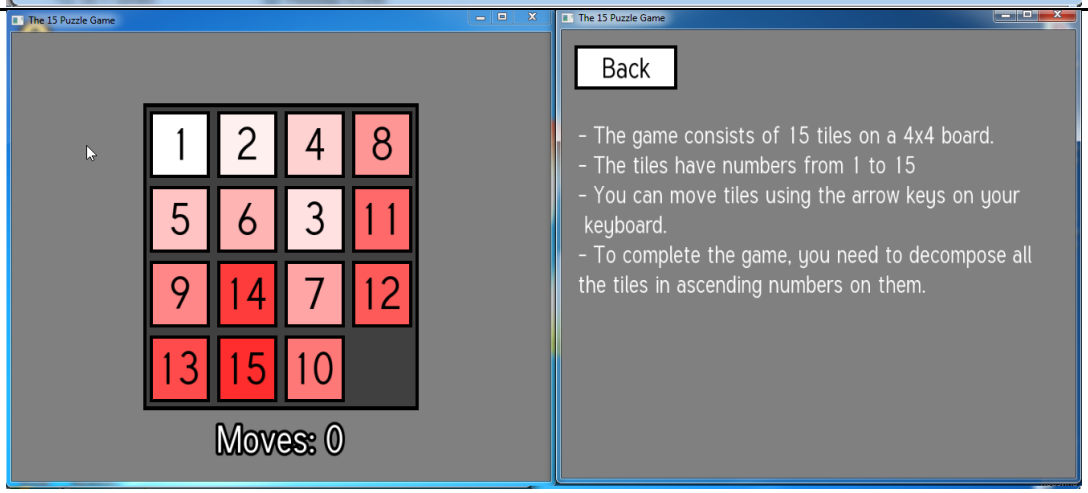
№	Результат
1	

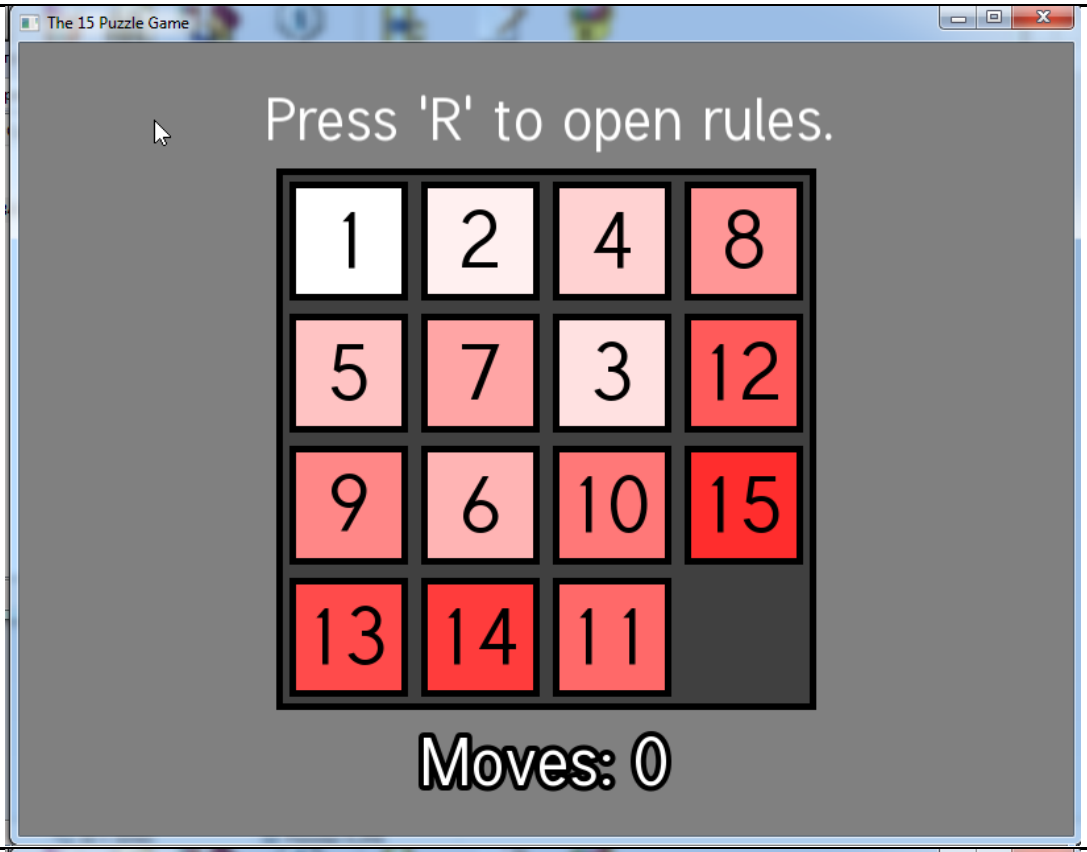
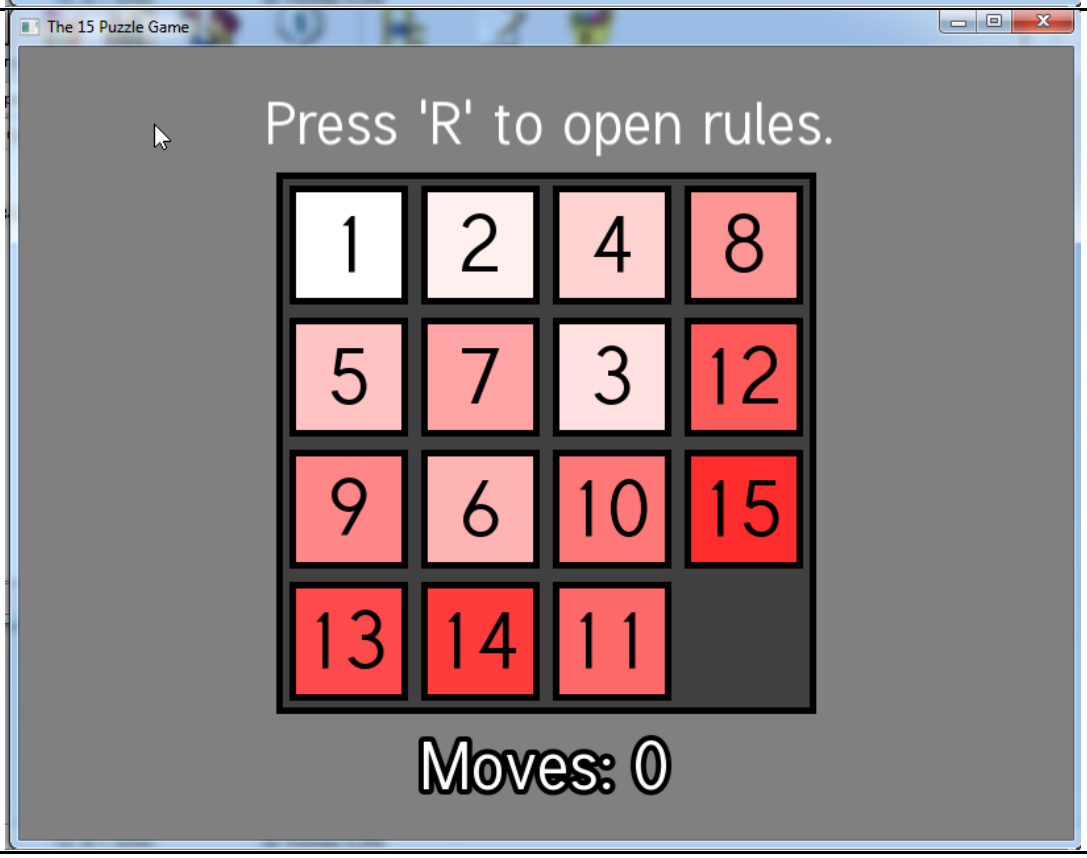
2	 The 15 Puzzle Game The 15 puzzle game Start Game Rules Exit
3	 The 15 Puzzle Game The 15 puzzle game Start Game Rules Exit The 15 Puzzle Game Back - The game consists of 15 tiles on a 4x4 board. - The tiles have numbers from 1 to 15 - You can move tiles using the arrow keys on your keyboard. - To complete the game, you need to decompose all the tiles in ascending numbers on them.

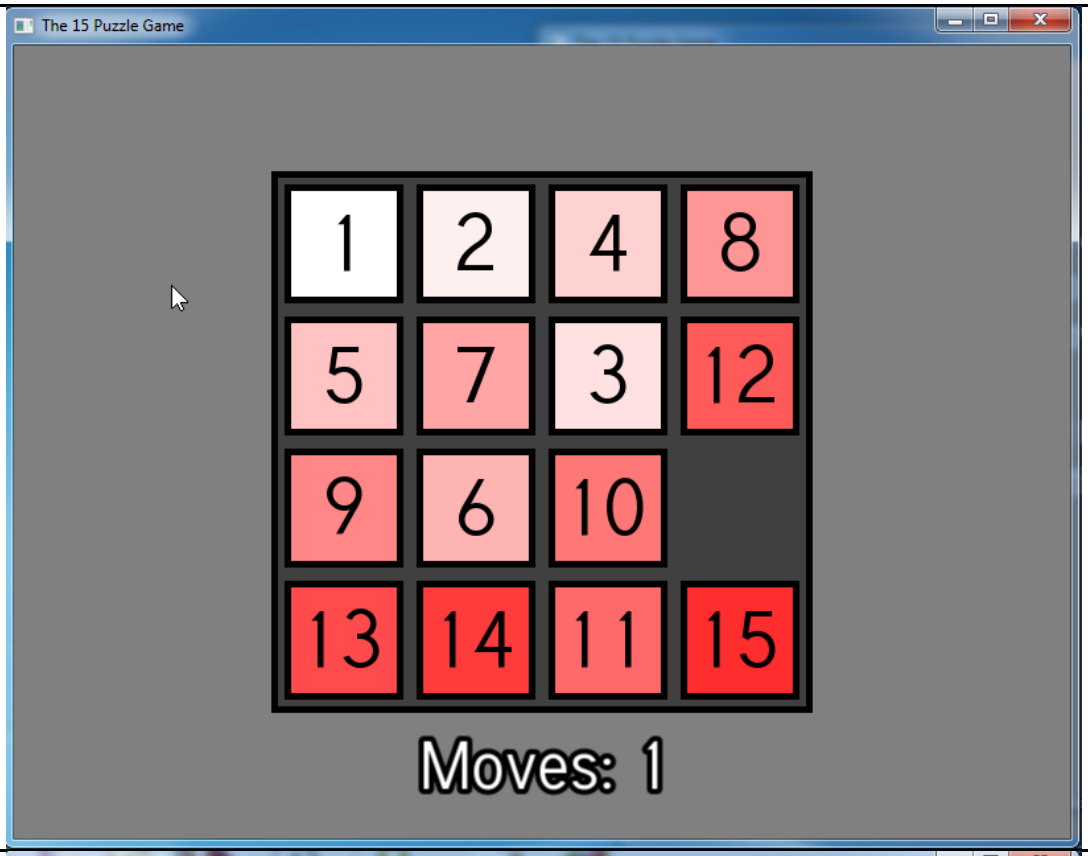
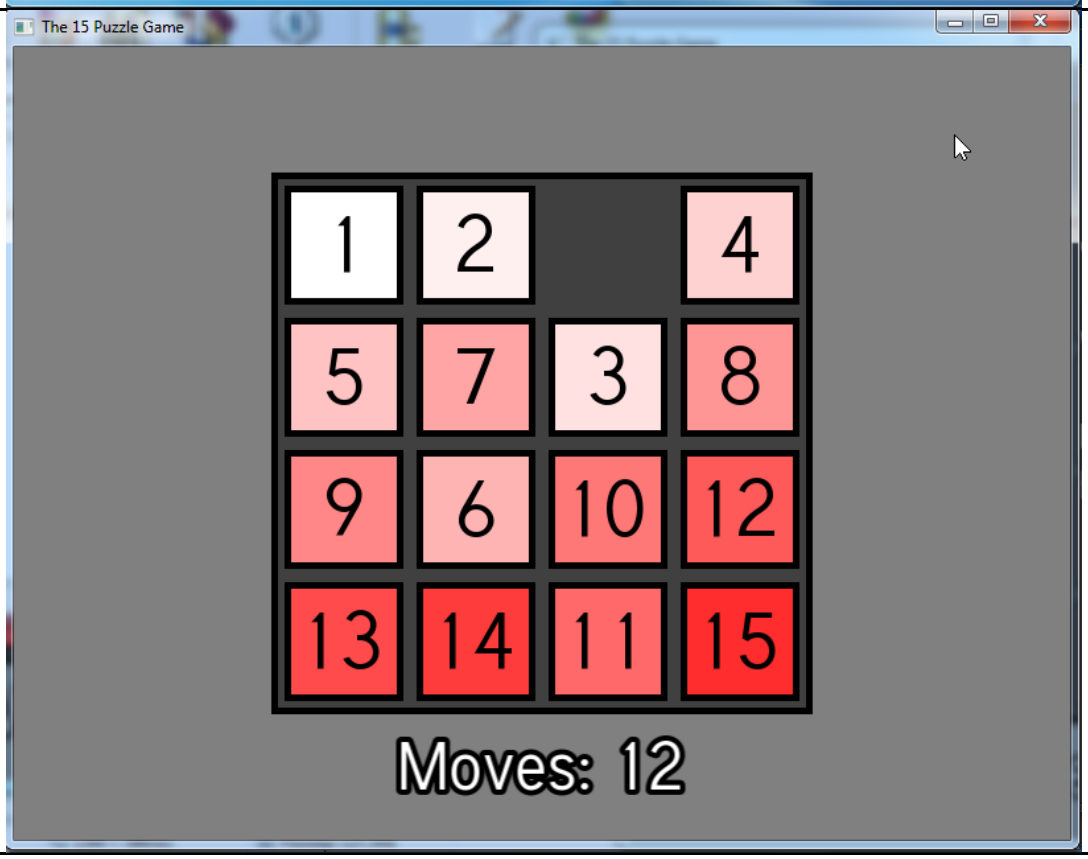
4	 The 15 Puzzle Game The 15 Puzzle Game Back - The game consists of 15 tiles on a 4x4 board. - The tiles have numbers from 1 to 15 - You can move tiles using the arrow keys on your keyboard. - To complete the game, you need to decompose all the tiles in ascending numbers on them.
5	 The 15 Puzzle Game The 15 Puzzle Game Back - The game consists of 15 tiles on a 4x4 board. - The tiles have numbers from 1 to 15 - You can move tiles using the arrow keys on your keyboard. - To complete the game, you need to decompose all the tiles in ascending numbers on them.

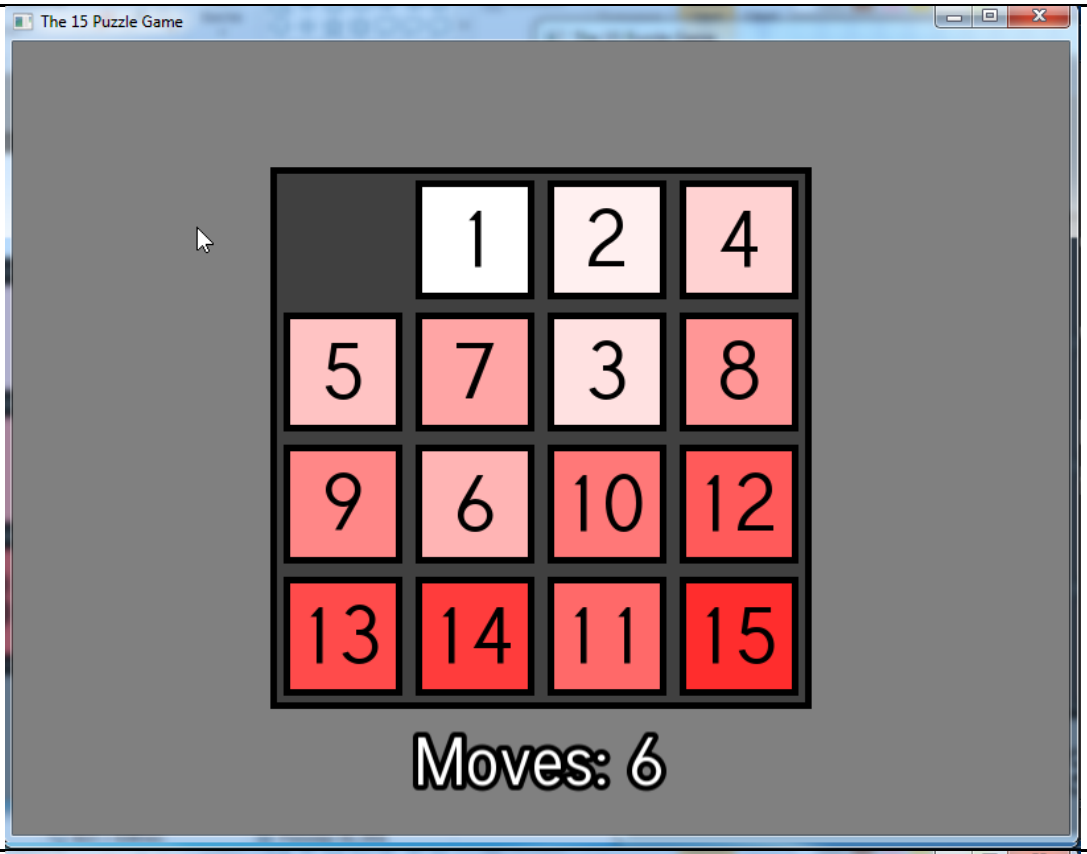
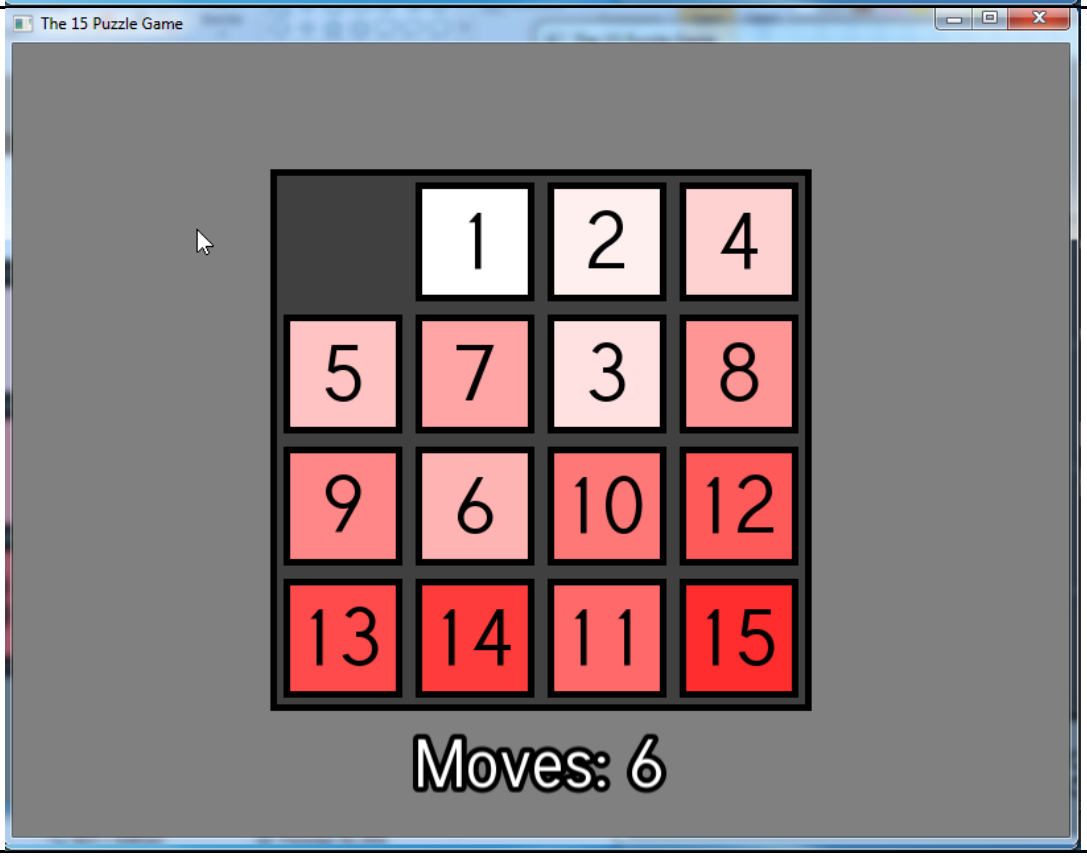
6	 The 15 Puzzle Game Back - The game consists of 15 tiles on a 4x4 board. - The tiles have numbers from 1 to 15 - You can move tiles using the arrow keys on your keyboard. - To complete the game, you need to decompose all the tiles in ascending numbers on them. The 15 Puzzle Game The 15 puzzle game Start Game Rules Exit
7	 The 15 Puzzle Game The 15 Puzzle Game The 15 puzzle game Start Game Rules Exit

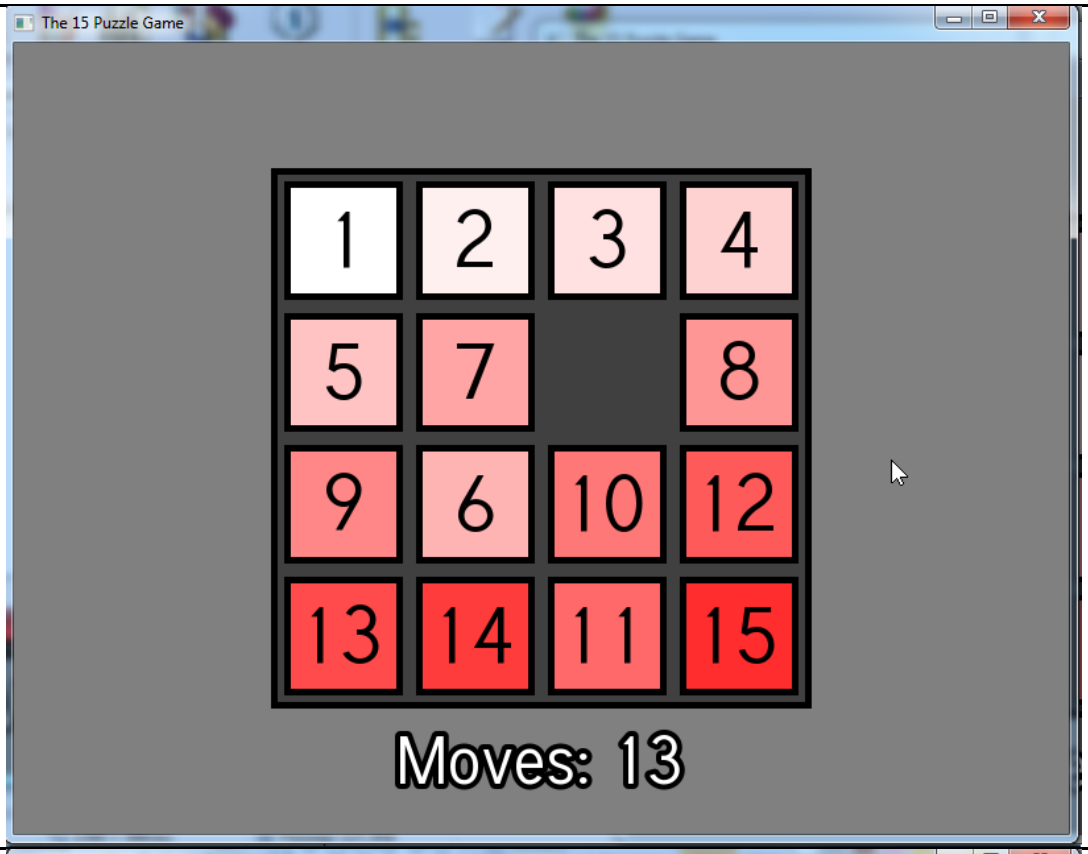
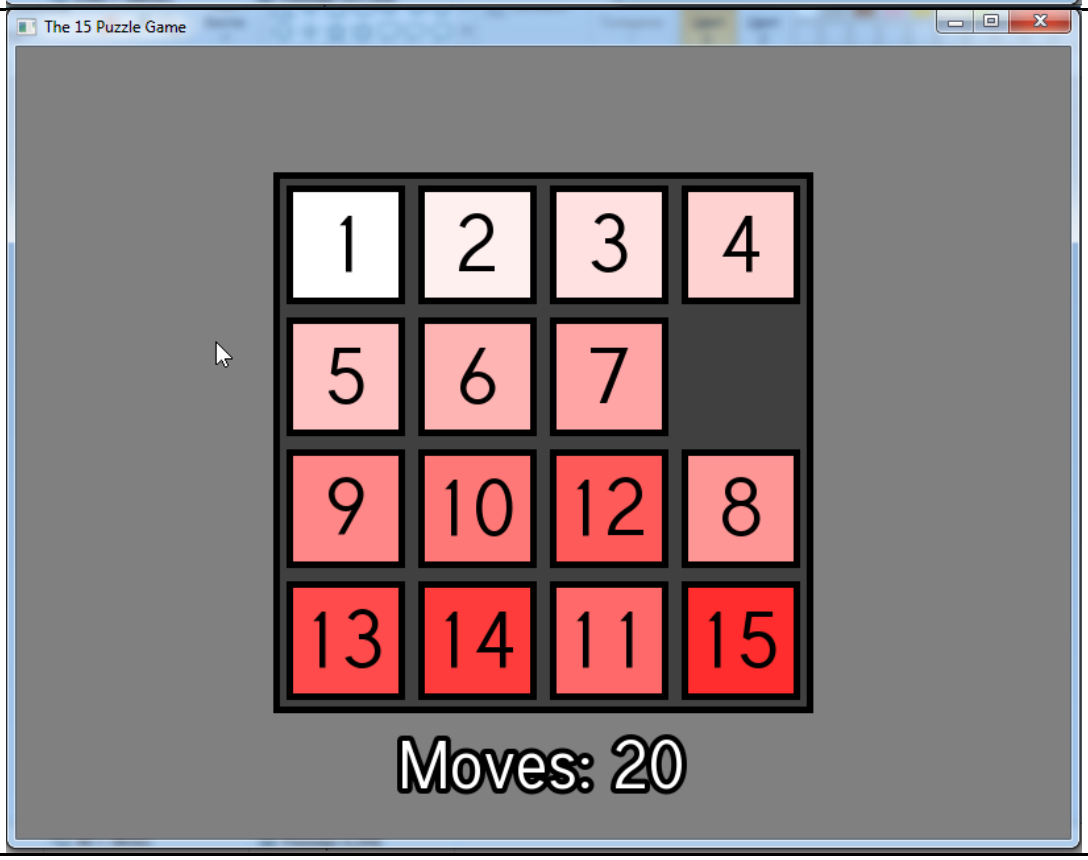
8	 The 15 Puzzle Game The 15 puzzle game Start Game Rules Exit The 15 Puzzle Game Press 'R' to open rules. 1 2 4 8 5 6 3 11 9 14 7 12 13 15 10 Moves: 0
9	 The 15 Puzzle Game Press 'R' to open rules. 1 2 4 8 5 7 3 12 9 6 10 15 13 14 11 Moves: 0

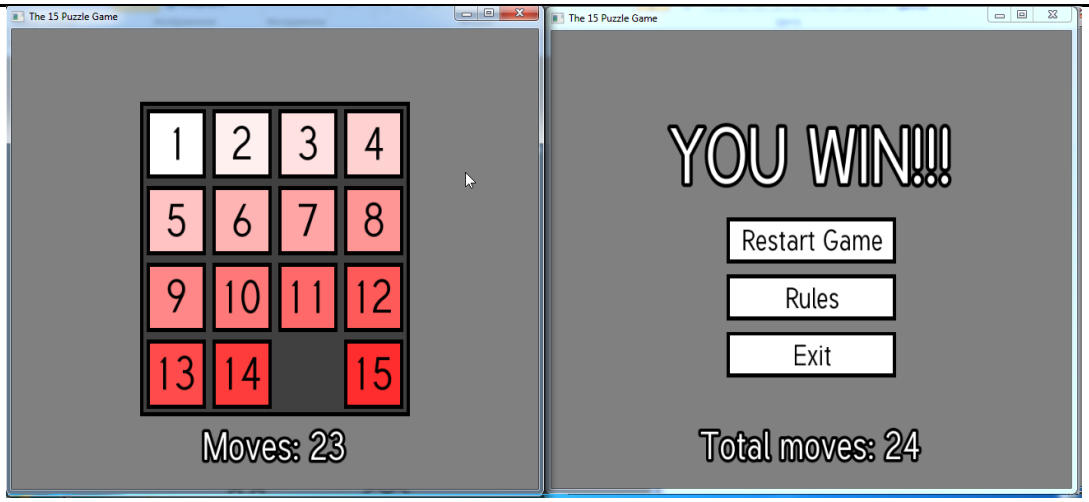
10	 The 15 Puzzle Game Press 'R' to open rules. 1 2 4 8 5 7 3 12 9 6 10 15 13 14 11 Moves: 0
11	 The 15 Puzzle Game 1 2 4 8 5 6 3 11 9 14 7 12 13 15 10 Moves: 0 The 15 Puzzle Game Back - The game consists of 15 tiles on a 4x4 board. - The tiles have numbers from 1 to 15 - You can move tiles using the arrow keys on your keyboard. - To complete the game, you need to decompose all the tiles in ascending numbers on them.

12	 The 15 Puzzle Game Press 'R' to open rules. 1 2 4 8 5 7 3 12 9 6 10 15 13 14 11 Moves: 0
13	 The 15 Puzzle Game Press 'R' to open rules. 1 2 4 8 5 7 3 12 9 6 10 15 13 14 11 Moves: 0

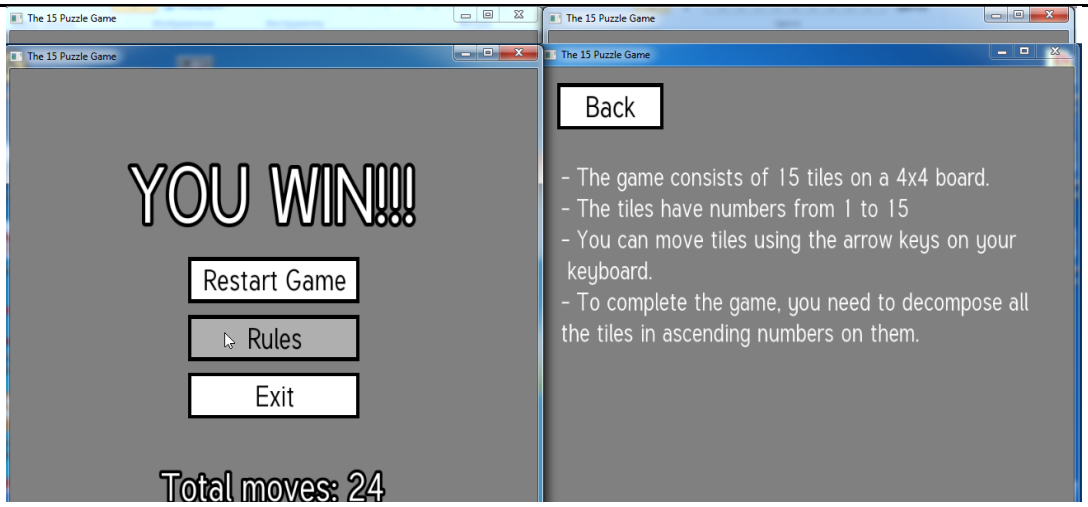
14	 The 15 Puzzle Game 1 2 4 8 5 7 3 12 9 6 10 13 14 11 15 Moves: 1
15	 The 15 Puzzle Game 1 2 4 5 7 3 8 9 6 10 12 13 14 11 15 Moves: 12

16	 The 15 Puzzle Game 1 2 4 5 7 3 8 9 6 10 12 13 14 11 15 Moves: 6
17	 The 15 Puzzle Game 1 2 4 5 7 3 8 9 6 10 12 13 14 11 15 Moves: 6

18	 The 15 Puzzle Game 1 2 3 4 5 7 8 9 6 10 12 13 14 11 15 Moves: 13
19	 The 15 Puzzle Game 1 2 3 4 5 6 7 9 10 12 8 13 14 11 15 Moves: 20

20	 The 15 Puzzle Game 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 Moves: 23 The 15 Puzzle Game YOU WIN!!! Restart Game Rules Exit Total moves: 24
21	 The 15 Puzzle Game YOU WIN!!! Restart Game Rules Exit Total moves: 24

22	 The 15 Puzzle Game YOU WIN!!! Restart Game Rules Exit Total moves: 24
23	 The 15 Puzzle Game YOU WIN!!! Restart Game Rules Exit Total moves: 24 The 15 Puzzle Game 1 6 2 4 5 3 8 11 9 10 7 12 13 14 15 Moves: 0

24	 The 15 Puzzle Game YOU WIN!!! Restart Game Rules Exit Total moves: 24 Back - The game consists of 15 tiles on a 4x4 board. - The tiles have numbers from 1 to 15 - You can move tiles using the arrow keys on your keyboard. - To complete the game, you need to decompose all the tiles in ascending numbers on them.
25	 The 15 Puzzle Game YOU WIN!!! Restart Game Rules Exit Total moves: 24

Заключение

В процессе прохождения практики выполнены следующие задачи:

– проведен сбор и анализ требований заказчика к программному средству;

– выполнена формализация выбранной предметной области;

– разработан проект программного средства;

– реализован проект программного средства;

– разработаны тесты и проведено тестирование программного средства;

– разработана и оформлена отчетная документация.

Таким образом, цели учебно-технологической (проектно-технологической) практики достигнуты.

Список использованных источников

1. Язык программирования C++. Лекции и упражнения / Прата Стивен. – М. : ООО “И. Д. Вильямс”, 2012 г. – 1248 с.

2. C++ для чайников, 6-е изд: Пер. с англ. / Стефан Р. Дэвис. – М. : ООО “И. Д. Вильямс”, 2010. – 336 с.

3. Документация по языку C++. Электронный ресурс. – Режим доступа: https://learn.microsoft.com/ru-ru/cpp/cpp/?view=msvc-160 [дата обращения 23.07.2023].

4. Обзор графических библиотек C++. Электронный ресурс. – Режим доступа: https://tproger.ru/digest/cpp-best-gui/ [дата обращения 26.07.2023].

5. Библиотека SFML. Электронный ресурс. – Режим доступа: https://www.sfml-dev.org/ [дата обращения 27.07.2023].