**НИЖЕГОРОДСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ**

**УНИВЕРСИТЕТ им. Р.Е. Алексеева**

Кафедра «Компьютерные технологии в проектировании и производстве»

УТВЕРЖДАЮ

*Зав. Кафедрой*

З А Д А Н И Е

на курсовое проектирование

Студент \_\_\_Лисин Илья Вадимович, гр. 13-СБК\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Тема курсового проекта \_\_Разработка игровой программы «Тетрис»\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Исходные данные к проекту \_Игровая программа должна обеспечивать:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

1.Реализацию игры в «Тетрис»\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

2.Рисование игрового поля\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

3.Рисование фигуры\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

4.Реализацию и обработку нажатия клавиш\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

5. Интуитивно понятный пользовательский интерфейс\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

6.Простоту расширения и модернизации\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Содержание графического материала:

чертежи: 1. \_Диаграмма прецедентов игры\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

2. Модель предметной области\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

3. Диаграмма классов\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

4. Диаграмма последовательностей\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

5. Диаграмма деятельностей\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Содержание пояснительной записки \_Введение, техническое задание, краткая\_\_\_\_\_\_\_\_\_\_\_ характеристика модели прецедентов, архитектура системы, детальная проработка\_\_\_\_\_\_\_ главного прецедента использования, разработка диаграммы последовательностей\_\_\_\_\_\_\_ игрового цикла, диаграмма деятельностей процесса игры, заключение, список литературы, приложение\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Основная рекомендуемая литература: 1. Логанов С.В., Язык UML и основы объектно-ориентированного проектирования ИС: учеб. Пособие / С.В. Логанов; Нижегород. гос.\_\_ техн.ун-т. – Н. Новгород, 2010. –139 с.\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

2. Гамма, Э. Приемы объектно-ориентированного проектирования. Паттерны\_\_\_\_\_\_\_\_\_\_\_ проектирования. / Э. Гамма, Р. Хелм, Р. Джонсон, Дж. Влиссидес. – СПб: «Питер», 2010 – 366 с.\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

3.Эккель Б., Философия С++. Введение в стандартный С++. 2-е изд.-СПБ: Питер, 2004 – 572 с.\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

*Руководитель* \_Логанов Сергей Викторович\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

*Консультанты* \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

" . . . " . . . . . . . . . . . . 2015 г. *Студент*

Федеральное агентство по образованию

Государственное образовательное учреждение

Высшего профессионального образования

НИЖЕГОРОДСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

Кафедра «Компьютерные технологии в проектировании и производстве»

Заведующий кафедрой Моругин С. Л*.*

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ (*фамилия, и., о.*)

(*подпись*)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(*дата*)

Разработка игровой программы «Тетрис»

**ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

К курсовой работе

**КОНСУЛЬТАНТЫ: РУКОВОДИТЕЛЬ:**

1. По\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Логанов С.В.

(*подпись*) (*фамилия, и., о.*)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ (*дата*)

(*подпись*) (*фамилия, и., о.*)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(*дата*)

2. По\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ **СТУДЕНТ**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Лисин И.В.

(*подпись*) (*фамилия, и., о.*) (*подпись*) (*фамилия, и., о.*)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 13-СБК

(*дата*) (*дата*) (группа или шифр)

**3**. По\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_ Проект защищен\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(*подпись*) (*фамилия, и., о.*) Протокол №\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ ­­­­ ­\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ С оценкой\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(*дата*)

РЕЦЕНЗЕНТ

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_

(*подпись*) (*фамилия, и., о.*)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(*дата*)

**2015**

**Содержание**

Изм.

Лист

№ докум.

Подпись

Дата

Лист

1

КР-НГТУ-09.03.02-13СБК-05

Разраб.

Лисин И.В.

Провер.

Логанов С.В

Реценз.

Н. Контр.

Утверд.

Пояснительная записка

Лит.

Листов

35

Кафедра КТПП

[1 Введение](#_Toc440363440) 2

[1 Техническое задание](#_Toc440363441) 3

[2 Краткая характеристика модели прецедентов](#_Toc440363442) 4

[4 Архитектура системы](#_Toc440363444) 6

[5 Детальная проработка главного прецедента использования](#_Toc440363445) 8

[6 Разработка диаграммы последовательностей игрового цикла](#_Toc440363446) 10

[7 Диаграмма деятельностей процесса игры](#_Toc440363450) 14

[8 Заключение](#_Toc440363451) 15

[Список литературы](#_Toc440363452) 16

[ПРИЛОЖЕНИЕ](#_Toc440363453) 17

[Код программы](#_Toc440363454) 17

[main.cpp](#_Toc440363455) 17

[tetrisview.h](#_Toc440363456) 17

[tetrisview.cpp](#_Toc440363457) 18

[game.h](#_Toc440363458) 20

[game.cpp](#_Toc440363459) 21

[figure.h](#_Toc440363460) 23

[figure.cpp](#_Toc440363461) 25

[controller.h](#_Toc440363462) 26

[controller.cpp](#_Toc440363463) 29

[board.h](#_Toc440363464) 31

[board.cpp](#_Toc440363465) 32

[figureview.h](#_Toc440363466) 34

[figureview.cpp](#_Toc440363467) 34

# 1 Введение

Изм.

Лист

№ докум.

Подпись

Дата

Лист

2

КР-НГТУ-09.03.02-13СБК-05

Целью данной курсовой работы является разработка игровой программы «Тетрис».

История игры «Тетрис» начинается в июне 1985 года. «Тетрис» был изобретен Алексеем Пажитновым, а затем был интегрирован на ПК IBM Вадимом Герасимовым. После чего ига «Тетрис» начала распространяться по всей Москве, а затем уже и по всему миру. Сначала она была доставлена в Венгрию, где венгерские программисты интегрировали «Тетрис» для Apple II и Commodore 64.

Игра была замечена в мире, и несколько представителей крупных компаний, обращались к автору «Тетриса», чтобы купить права на распространение игры. Алексей подписывает контракт с Mirrorsoft UK и Spectrum Holobyte, предоставляя им права на компьютерным версии «Тетрис». После того, как первые копии «Тетриса» для домашних компьютеров были преданны, игра приобрела популярность среди населения и стала самой продаваемой компьютерной игрой в Англии и США в 1988 году.

Сейчас «Тетрис» и его различные модификации разрабатывается многими компаниями. Это означает, что он остается популярным в настоящее время.

# 1 Техническое задание

Изм.

Лист

№ докум.

Подпись

Дата

Лист

3

КР-НГТУ-09.03.02-13СБК-05

Требуется разработать игровую программу «Тетрис». Он должен:

* Создавать игровое поле
* Обладать интуитивно понятным интерфейсом
* Случайным образом генерировать фигуры разного цвета
* Очищать заполненные ряды
* Позволять пользователю менять положение фигурки в пространстве и вращать ее против часовой стрелки
* Обеспечить возможность легкого расширения для добавления новых фигур

# 2 Краткая характеристика модели прецедентов

Изм.

Лист

№ докум.

Подпись

Дата

Лист

4

КР-НГТУ-09.03.02-13СБК-05

Назначение программы - развлечение играющих, совершенствование их координации и логического мышления.

Программа должна осуществлять вывод на экран случайным образом последовательное падение различных фигур. Входными данными являются ввод скорости движения фигур сверху вниз, сдвиг фигур по горизонтали вправо и влево, а также поворот фигур против часовой стрелки.

# 

Рисунок 1 - Диаграмма прецедентов использования

Прецеденты:

* Играть. Основной прецедент, выполняющий игровой цикл.
* Сдвиг фигуры. Прецедент, выполняющий сдвиг фигуры вправо/влево.
* Вращение фигуры. Прецедент, выполняющий вращение фигуры против часовой стрелки.
* Ускорить падение фигуры. Прецедент, выполняющий ускорение падения фигуры.

Актер – пользователь (игрок), непосредственно взаимодействующий с системой. Он может играть и манипулировать фигурой.

**3 Модель предметной области**

Изм.

Лист

№ докум.

Подпись

Дата

Лист

5

КР-НГТУ-09.03.02-13СБК-05

Случайные фигурки падают сверху в прямоугольный стакан (доска). В полёте игрок может поворачивать фигурку и двигать её по горизонтали. Также можно «сбрасывать» фигурку, то есть ускорять её падение. Фигурка падает, пока не наткнётся на другую фигурку либо на дно стакана. Если при этом заполнился горизонтальный ряд, он пропадает и всё, что выше него, опускается на одну клетку. Игра заканчивается, когда новая фигурка не может поместиться в стакан. Задача игрока — заполнять ряды, не заполняя сам стакан (по вертикали) как можно дольше.

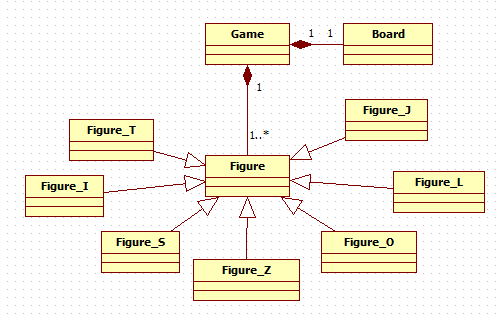


Рисунок 2 - Модель предметной области

Game – объект, содержащий логику и основной игровой цикл. Он осуществляет начальную инициализацию игры, владеет информацией о доске и активной фигуре, отдает команды по перемещению фигуры, связывает фигуру с доской. Также он на каждом шаге производит обработку столкновений активной фигуры с элементами доски и ее границами.

Board – доска/игровое поле. Она знает свои размеры и что находится в каждой ячейке. Доска может очистить себя, установить фигуру на поле, проверить заполненные линии и удалить их.

Figure – объект фигуры, являющийся родителем для различных типов фигур. Фигура может генерировать различные типы фигур. Каждый тип фигуры знает себя и свой размер. Фигура знает свои координаты, может повернуться против часовой стрелки и сообщить свой тип.

# 4 Архитектура системы

Изм.

Лист

№ докум.

Подпись

Дата

Лист

6

КР-НГТУ-09.03.02-13СБК-05

Основной шаблон, который будет использоваться в коде игры — MVC (Model-View-Controller). Основная цель применения этого шаблона состоит в отделении бизнес-логики (модели) от её визуализации (представления, вида). За счет такого разделения повышается возможность повторного использования. Наиболее полезно применение данной концепции в тех случаях, когда пользователь должен видеть те же самые данные одновременно в различных контекстах и/или с различных точек зрения. В частности, выполняются следующие задачи:

* К одной модели можно присоединить несколько видов, при этом не затрагивая реализацию модели. Например, некоторые данные могут быть одновременно представлены в виде электронной таблицы, гистограммы и круговой диаграммы.
* Не затрагивая реализацию видов, можно изменить реакции на действия пользователя (нажатие мышью на кнопке, ввод данных), для этого достаточно использовать другой контроллер.
* Ряд разработчиков специализируется только в одной из областей: либо разрабатывают графический интерфейс, либо разрабатывают бизнес-логику. Поэтому возможно добиться того, что программисты, занимающиеся разработкой бизнес-логики (модели), вообще не будут осведомлены о том, какое представление будет использоваться.

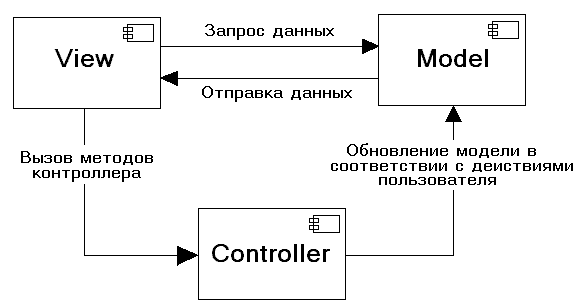
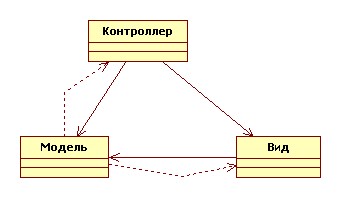


Рисунок 3 - Диаграмма логических компонентов шаблона MVC

В архитектуре MVC модель предоставляет данные и правила логики, представление отвечает за пользовательский интерфейс, а контроллер обеспечивает взаимодействие между моделью и представлением.



Изм.

Лист

№ докум.

Подпись

Дата

Лист

7

КР-НГТУ-09.03.02-13СБК-05

Рисунок 4 - Диаграмма классов MVC

Модель игры написана на языке высокого уровня С++. Отображение игры использует кроссплатформенную библиотеку QT - QWidget.

Qt — кроссплатформенный инструментарий разработки ПО на языке программирования C++.

Qt позволяет запускать написанное с его помощью ПО в большинстве современных операционных систем путём простой компиляции программы для каждой ОС без изменения исходного кода. Включает в себя все основные классы, которые могут потребоваться при разработке прикладного программного обеспечения, начиная от элементов графического интерфейсами заканчивая классами для работы с сетью, базами данных и XML. Qt является полностью объектно-ориентированным, легко расширяемым и поддерживающим технику компонентного программирования.

QWidget довольная простая и удобная в использовании библиотека для написания оконных приложений.

# 5 Детальная проработка главного прецедента использования

Изм.

Лист

№ докум.

Подпись

Дата

Лист

8

КР-НГТУ-09.03.02-13СБК-05

Сценарий прецедента:

* 1. Краткое описание

Пользователь запускает игру. Появляется игровое поле и начинает падать случайно сгенерированная фигура. Пользователь может ее переместить, повернуть и ускорить. По заполнению ряда, он удаляется. Если фигуре падать некуда, игра перезапускается.

* 1. Актеры
     1. Пользователь
  2. Выполнение
     1. Пользователь запускает игру
     2. Появляется игровое поле и случайно сгенерированная фигура вверху игрового поля
     3. Фигура начинает падение
     4. Пользователь манипулирует фигурой с помощью клавиш и ставит ее на нужное место на игровом поле

1.3.4а Нажата клавиша «стрелка влево» - фигура перемещается на одно деление влево

1.3.4б Нажата клавиша «стрелка вправо» - фигура перемещается на одно деление вправо

1.3.4в Нажата клавиша «стрелка вверх» - фигура поворачивается на 90 градусов против часовой стрелки

1.3.4г Нажата клавиша «стрелка вниз» - фигура ускоряется

1.3.4д Нажата клавиша «пробел» - ставится пауза. Повторное нажатие возобновляет игру

* + 1. Проверка заполненных линий

1.3.5а Если линия заполнена, она удаляется и содержимое игрового поля смещается на одно деление вниз

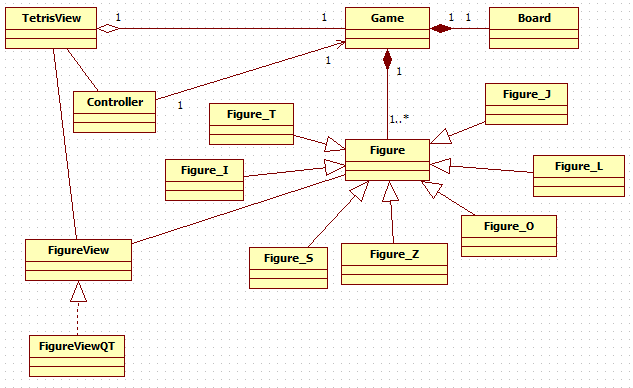
1.3.5б Если линия не заполнена, продолжение игры

* + 1. Генерируется новая фигура и повторяются пункты 1.3.3 – 1.3.5
    2. Проверка конца игры

1.3.7а Игра окончена, если случайно сгенерированной фигуре некуда падать

1.3.7а.1 если игра окончена, переход к пункту 1.3.2

На рисунке 4 представлена окончательная диаграмма классов с использованием шаблона MVC.



Изм.

Лист

№ докум.

Подпись

Дата

Лист

9

КР-НГТУ-09.03.02-13СБК-05

Рисунок 5 - Диаграмма классов

TetrisView – класс графического интерфейса. Он владеет информацией об игре и связан с контроллером. Он отвечает за рисование доски и фигуры, обрабатывает нажатия клавиш и сообщает об этом контроллеру.

Доска и фигура являются двумерными числовыми массивами, некими каркасами. Элементами доски являются цифры, каждая из которых соответствует либо границе, либо пустой ячейке, либо занятой частью определенной фигуры ячейкой. TetrisView «конвертирует» эти числовые массивы в графические элементы (блоки, квадраты, круги и тд) и выводит на экран.

Controller – класс, обеспечивающий взаимодействие между моделью и отображением. TetrisView посылает ему информацию о нажатой клавише, контроллер используя эту информацию вызывает соответствующие методы игры, и ее состояние изменяется.

FigureView – внешнее представление фигуры, абстрактный базовый класс, отвечающий за рисование фигуры. Он имеет информацию о ней.

FigureViewQT – конкретная реализация FigureView, рисующий фигуру средствами QT.

Таким образом, суть игры отделена от ее внешнего представления, что дает возможность отобразить игру различными инструментами.

# 6 Разработка диаграммы последовательностей игрового цикла

Изм.

Лист

№ докум.

Подпись

Дата

Лист

10

КР-НГТУ-09.03.02-13СБК-05

## C:\Users\Ilya\Desktop\sequence.png

Рисунок 6 - Диаграмма последовательностей игрового цикла

## 

Изм.

Лист

№ докум.

Подпись

Дата

Лист

11

КР-НГТУ-09.03.02-13СБК-05

## C:\Users\Ilya\Desktop\refdostep.png

Рисунок 7 - Диаграмма последовательностей выполнения шага игры

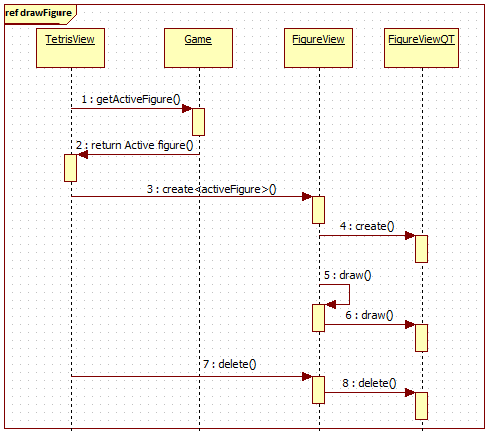
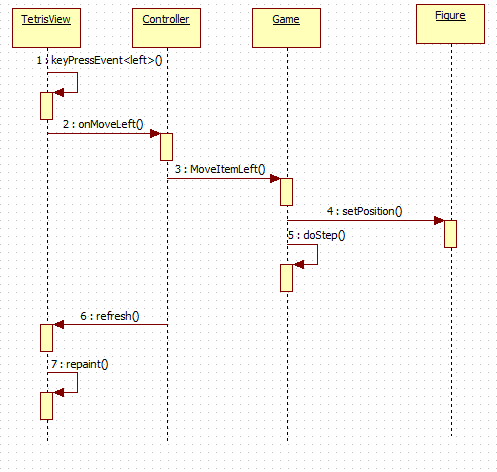


Рисунок 8 - Диаграмма последовательностей рисования фигуры

Следующая диаграмма последовательностей сдвига фигуры аналогична для сдвига влево и вправо.



Изм.

Лист

№ докум.

Подпись

Дата

Лист

12

КР-НГТУ-09.03.02-13СБК-05

Рисунок 9 - Диаграмма последовательностей сдвига фигуры

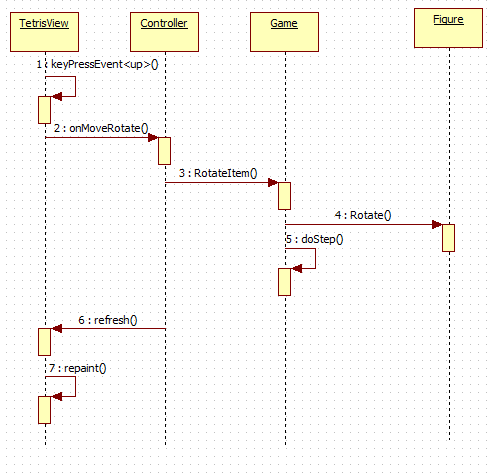
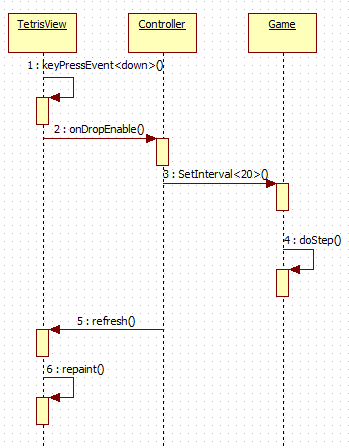


Рисунок 10. Диаграмма последовательностей вращения фигуры



Изм.

Лист

№ докум.

Подпись

Дата

Лист

13

КР-НГТУ-09.03.02-13СБК-05

Рисунок 11 - Диаграмма последовательностей ускоренного падения

# 7 Диаграмма деятельностей процесса игры

Рисунок 12 - Диаграмма деятельностей процесса игры

Изм.

Лист

№ докум.

Подпись

Дата

Лист

14

КР-НГТУ-09.03.02-13СБК-05

# 8 Заключение

Изм.

Лист

№ докум.

Подпись

Дата

Лист

15

КР-НГТУ-09.03.02-13СБК-05

Игровая программа «Тетрис» разработана по основным принципам ООП. Для реализации объектной модели использовался унифицированный язык моделирования UML. С помощью него была создана абстрактная модель системы и по этой модели написан код. Архитектура системы построена по шаблону MVC, что позволило отделить суть игры от внешнего представления, что дает возможность модернизировать программу, улучшить графические эффекты, использовать для отображения различные инструменты.

# Список литературы

Изм.

Лист

№ докум.

Подпись

Дата

Лист

16

КР-НГТУ-09.03.02-13СБК-05

1. Логанов С.В., Язык UML и основы объектно-ориентированного проектирования ИС: учеб. Пособие / С.В. Логанов; Нижегород. гос. техн.ун-т. – Н. Новгород, 2010. –139 с.
2. Гамма, Э. Приемы объектно-ориентированного проектирования. Паттерны проектирования. / Э. Гамма, Р. Хелм, Р. Джонсон, Дж. Влиссидес. – СПб: «Питер», 2010 – 366 с.
3. Эккель Б., Философия С++. Введение в стандартный С++. 2-е изд.-СПБ: Питер, 2004 – 572 с.
4. Vitaly Swipe, «Реализация MVC паттерна на примере создания сайта-визитки на PHP», http://habrahabr.ru/post/150267/
5. Zesetup, «Отношения классов — от UML к коду», http://habrahabr.ru/post/150041/
6. «QT Documentation» , http://doc.qt.io/
7. Dimorinny, «QT – окно и виджеты», http://habrahabr.ru/post/245245/
8. Джимми Уэйлс, «Тетрис», https://ru.wikipedia.org/wiki/Тетрис

# ПРИЛОЖЕНИЕ

Изм.

Лист

№ докум.

Подпись

Дата

Лист

17

КР-НГТУ-09.03.02-13СБК-05

## Код программы

### main.cpp

#include "tetrisview.h"

#include <QApplication>

int main(int argc, char \*argv[])

{

QApplication a( argc, argv );

TetrisView w;

w.show();

return a.exec();

}

### tetrisview.h

#ifndef TETRISVIEW\_H

#define TETRISVIEW\_H

#include <QWidget>

#include <game.h>

#include <controller.h>

#include "figureview.h"

class TetrisView : public QWidget

{

Q\_OBJECT

protected:

void paintEvent( QPaintEvent\* );

void keyPressEvent( QKeyEvent\* e );

void keyReleaseEvent( QKeyEvent\* e );

void drawBlock( int xPoints, int yPoints, int type, QPainter\* painter );

private:

static const int BLOCK\_SIZE=20;

Game game;

Controller\* controller;

FigureView\* fView;

public:

explicit TetrisView( QWidget\* parent = 0 );

~TetrisView();

void refresh();

};

#endif // TETRISVIEW\_H

### tetrisview.cpp

#include "tetrisview.h"

#include <QPainter>

Изм.

Лист

№ докум.

Подпись

Дата

Лист

18

КР-НГТУ-09.03.02-13СБК-05

#include <QKeyEvent>

TetrisView::TetrisView( QWidget\* parent ) : QWidget( parent )

{

controller = new Controller( &game, this, this );

controller->onStart();

resize(game.getMatrixWidth()\*BLOCK\_SIZE,game.getMatrixHeight()\*BLOCK\_SIZE );

}

TetrisView::~TetrisView(){}

void TetrisView::refresh()

{

repaint();

}

void TetrisView::paintEvent( QPaintEvent\* )

{

QPainter painter( this );

painter.fillRect( BLOCK\_SIZE, BLOCK\_SIZE, (game.getMatrixWidth()-2)\*BLOCK\_SIZE, (game.getMatrixHeight()-2)\*BLOCK\_SIZE, Qt::black );

for( int i = 0; i < game.getMatrixHeight();i++ )

{

for( int j = 0; j < game.getMatrixWidth(); j++ )

{

drawBlock( i, j, game.getType(i,j), &painter );

}

}

Figure \*f\_active = game.getFigure();

fView=new FigureViewQT(f\_active,this);

fView->draw();

delete fView;

}

void TetrisView::keyPressEvent( QKeyEvent\* e )

{

switch( e->key() )

{

case Qt::Key\_Left:

controller->onMoveLeft();

break;

case Qt::Key\_Right:

controller->onMoveRight();

break;

case Qt::Key\_Up:

controller->onRotate();

break;

case Qt::Key\_Down:

controller->onDropEnabled( true );

break;

case Qt::Key\_Space:

Изм.

Лист

№ докум.

Подпись

Дата

Лист

19

КР-НГТУ-09.03.02-13СБК-05

controller->onTogglePause();

break;

default:

QWidget::keyPressEvent( e );

}

}

void TetrisView::keyReleaseEvent( QKeyEvent\* e )

{

switch( e->key() )

{

case Qt::Key\_Down:

controller->onDropEnabled( false );

break;

default:

QWidget::keyReleaseEvent( e );

}

}

void TetrisView::drawBlock( int x, int y, int type , QPainter\* painter )

{

static const std::vector< QColor > COLOR\_TABLE = {

Qt::white,

Qt::yellow,

Qt::green,

Qt::red,

Qt::cyan,

Qt::magenta,

Qt::lightGray

Изм.

Лист

№ докум.

Подпись

Дата

Лист

20

КР-НГТУ-09.03.02-13СБК-05

};

if (type==0 || type == -1) return;

if( type <= 0 || COLOR\_TABLE.size() < type )

{

return;

}

//painter->drawRect(y\*BLOCK\_SIZE, x\*BLOCK\_SIZE, BLOCK\_SIZE, BLOCK\_SIZE);

painter->fillRect( y\*BLOCK\_SIZE+1, x\*BLOCK\_SIZE+1, BLOCK\_SIZE-1, BLOCK\_SIZE-1, COLOR\_TABLE[ type - 1 ] );

}

### game.h

#ifndef GAME\_H

#define GAME\_H

#include "figure.h"

#include "board.h"

#include <vector>

class Game

{

private:

Figure \*f\_active; //активная фигура

Board board;

bool gameOver;

int BottomTouchCounter;

bool isCollision(Figure f); //обработка столкновений

public:

Game();

void start();

int getMatrixHeight();

int getMatrixWidth();

int getType(int,int); // получить тип эелемента ячейки доски

Figure\* getFigure();

bool isGameOver();

void doStep();

void moveItemLeft();

void moveItemRight();

void rotateItem();

};

#endif // GAME\_H

### game.cpp

#include "game.h"

Game::Game(){}

void Game::start()

{

f\_active=NULL;

gameOver=false;

board.refresh();

}

bool Game::isCollision(Figure f)

Изм.

Лист

№ докум.

Подпись

Дата

Лист

21

КР-НГТУ-09.03.02-13СБК-05

{//i - height, j - width

for(int i=0;i<f.getSize();i++)

for(int j=0;j<f.getSize();j++)

if(f.getFigureMatrixElem(i,j)!=0 && board.getMatrixElem(f.getY()+i,f.getX()+j)!=0)

return true;

return false;

}

void Game::doStep()

{

board.Draw();

if (f\_active==NULL)

{

BottomTouchCounter=0;

f\_active=Figure::generateFigure();

f\_active->setPosition((board.getWidth()/2)-1,1);

if(isCollision(\*f\_active))

{

gameOver = true;

}

}

if(isGameOver()) return;

f\_active->setPosition(f\_active->getX(), f\_active->getY()+1);

if(!isCollision(\*f\_active))

{

BottomTouchCounter=0;

}

else

{

while(isCollision(\*f\_active))

f\_active->setPosition(f\_active->getX(),f\_active->getY()-1);

if( BottomTouchCounter > 2)

{

board.setFigure(\*f\_active);

delete f\_active;

f\_active=NULL;

}

else

{

BottomTouchCounter++;

}

}

board.checkSolidLine();

}

bool Game::isGameOver()

{

Изм.

Лист

№ докум.

Подпись

Дата

Лист

22

КР-НГТУ-09.03.02-13СБК-05

return gameOver;

}

int Game::getMatrixHeight()

{

return board.getHeight();

}

int Game::getMatrixWidth()

{

return board.getWidth();

}

int Game::getType(int i, int j)

{

return board.getMatrixElem(i,j);

}

Изм.

Лист

№ докум.

Подпись

Дата

Лист

23

КР-НГТУ-09.03.02-13СБК-05

Figure\* Game::getFigure()

{

return f\_active;

}

void Game::moveItemLeft()

{

if(f\_active==NULL) return;

f\_active->setPosition(f\_active->getX()-1, f\_active->getY());

if (isCollision(\*f\_active)) f\_active->setPosition(f\_active->getX()+1, f\_active->getY());

}

void Game::moveItemRight()

{

if(f\_active==NULL) return;

f\_active->setPosition(f\_active->getX()+1, f\_active->getY());

if (isCollision(\*f\_active)) f\_active->setPosition(f\_active->getX()-1, f\_active->getY());

}

void Game::rotateItem()

{

Figure f\_copy;

f\_copy=\*f\_active;

f\_copy.rotate();

if(isCollision(f\_copy)) return;

if(f\_active==NULL) return;

f\_active->rotate();

}

### figure.h

#ifndef FIGURE\_H

#define FIGURE\_H

#include <vector>

#include <cstdlib>

#include <time.h>

class Figure

{

protected:

int size;

Изм.

Лист

№ докум.

Подпись

Дата

Лист

24

КР-НГТУ-09.03.02-13СБК-05

int blockType;

std::vector<std::vector<int> > figureMatrix;

private:

const static int COUNT\_OF\_FIGURE=7;

int x,y; // координаты фигуры

public:

Figure();

void setPosition(int i,int j); // установить координаты фигуры

int getX();

int getY();

int getSize();

void rotate();

int getBlockType(int i,int j); // получить тип блока

static Figure\* generateFigure();

int getFigureMatrixElem(int i,int j);

};

class Figure\_I: public Figure

{

public:

Figure\_I();

};

class Figure\_J: public Figure

{

public:

Figure\_J();

};

class Figure\_L:public Figure

{

public:

Figure\_L();

};

class Figure\_O:public Figure

{

public:

Figure\_O();

};

class Figure\_S:public Figure

{

public:

Figure\_S();

};

class Figure\_T: public Figure

{

public:

Figure\_T();

};

class Figure\_Z: public Figure

{

public:

Figure\_Z();

};

#endif // FIGURE\_H

Изм.

Лист

№ докум.

Подпись

Дата

Лист

25

КР-НГТУ-09.03.02-13СБК-05

### figure.cpp

#include "figure.h"

Figure::Figure()

{

x=5;

y=1;

}

Figure\_I::Figure\_I()

{

Figure();

figureMatrix={

{0,1,0,0},

{0,1,0,0},

{0,1,0,0},

{0,1,0,0} };

size=4;

blockType=1;

}

Figure\_J::Figure\_J()

{

figureMatrix={

{0,2,0},

{0,2,0},

{2,2,0} };

size=3;

blockType=2;

}

Изм.

Лист

№ докум.

Подпись

Дата

Лист

26

КР-НГТУ-09.03.02-13СБК-05

Figure\_L::Figure\_L()

{

figureMatrix={

{0,3,0},

{0,3,0},

{0,3,3} };

size=3;

blockType=3;

}

Figure\_O::Figure\_O()

{

figureMatrix={

{4,4},

{4,4} };

size=2;

blockType=4;

}

Figure\_S::Figure\_S()

{

figureMatrix={

{0,0,0},

{0,5,5},

{5,5,0} };

size=3;

blockType=5;

}

Figure\_T::Figure\_T()

{

figureMatrix={

{0,0,0},

{0,6,0},

{6,6,6} };

size=3;

blockType=6;

}

Figure\_Z::Figure\_Z()

{

figureMatrix={

{0,0,0},

{7,7,0},

{0,7,7} };

size=3;

blockType=7;

}

void Figure::rotate()

Изм.

Лист

№ докум.

Подпись

Дата

Лист

27

КР-НГТУ-09.03.02-13СБК-05

{

std::vector<std::vector<int>> buf;

buf=figureMatrix;

for(int i=0;i<figureMatrix.size();i++)

for(int j=0; j<figureMatrix.size();j++)

figureMatrix[i][j]=buf[j][figureMatrix.size()-i-1];

}

void Figure::setPosition(int \_x,int \_y)

{

x=\_x;

y=\_y;

}

int Figure::getX()

{

return x;

}

int Figure::getY()

{

return y;

}

int Figure::getSize()

{

return size;

}

int Figure::getBlockType(int i, int j)

{

return figureMatrix[i][j];

}

Figure\* Figure::generateFigure()

{

srand(time(NULL));

int p;

p=random()%COUNT\_OF\_FIGURE + 1;

switch(p)

{

case 1: {Figure\_I\* figure=new Figure\_I(); return figure;}

case 2: {Figure\_J\* figure=new Figure\_J(); return figure;}

case 3: {Figure\_L\* figure=new Figure\_L(); return figure;}

case 4: {Figure\_O\* figure=new Figure\_O(); return figure;}

case 5: {Figure\_S\* figure=new Figure\_S(); return figure;}

case 6: {Figure\_T\* figure=new Figure\_T(); return figure;}

case 7: {Figure\_Z\* figure=new Figure\_Z(); return figure;}

Изм.

Лист

№ докум.

Подпись

Дата

Лист

28

КР-НГТУ-09.03.02-13СБК-05

default: {Figure\_I\* figure=new Figure\_I(); return figure;}

}

}

int Figure::getFigureMatrixElem(int i, int j)

{

return figureMatrix[i][j];

}

### controller.h

#ifndef TETRISCONTROLLER\_H

#define TETRISCONTROLLER\_H

#include <QObject>

#include <QTimer>

class Game;

class TetrisView;

class Controller : QObject

{

Q\_OBJECT

public:

Controller( Game\* c\_game, TetrisView\* c\_view, QObject\* parent = 0 );

public slots:

void onStart();

void onPause();

void onResume();

void onStep();

void onMoveLeft();

void onMoveRight();

void onRotate();

void onDropEnabled( bool enabled );

void onTogglePause();

private:

void onAction( void ( Game::\*action )() );

Game\* game;

TetrisView\* view;

QTimer\* timer;

};

#endif // TETRISCONTROLLER\_H

### controller.cpp

#include "controller.h"

Изм.

Лист

№ докум.

Подпись

Дата

Лист

29

КР-НГТУ-09.03.02-13СБК-05

#include "game.h"

#include "tetrisview.h"

#include <QDebug>

Controller::Controller( Game\* c\_game , TetrisView\* c\_view, QObject\* parent ) :

QObject( parent ), game( c\_game ), view( c\_view )

{

timer = new QTimer();

timer->setInterval(500);

connect( timer, SIGNAL( timeout() ), SLOT( onStep() ) );

}

void Controller::onStart()

{

game->start();

timer->start();

}

void Controller::onStep()

{

game->doStep();

view->refresh();

if(game->isGameOver())

{

timer->stop();

onStart();

}

}

void Controller::onPause()

{

timer->stop();

}

void Controller::onResume()

{

timer->start(500);

}

void Controller::onMoveLeft()

{

onAction( &Game::moveItemLeft );

}

void Controller::onMoveRight()

{

onAction( &Game::moveItemRight );

}

void Controller::onRotate()

{

onAction( &Game::rotateItem );

Изм.

Лист

№ докум.

Подпись

Дата

Лист

30

КР-НГТУ-09.03.02-13СБК-05

}

void Controller::onDropEnabled( bool enabled )

{

if(enabled) timer->setInterval(20);

else

timer->setInterval(500);

}

Изм.

Лист

№ докум.

Подпись

Дата

Лист

31

КР-НГТУ-09.03.02-13СБК-05

void Controller::onTogglePause()

{

timer->isActive() ? onPause() : onResume();

}

void Controller::onAction( void ( Game::\*action )() )

{

( game->\*action )();

view->refresh();

}

### board.h

#ifndef BOARD\_H

#define BOARD\_H

#include <iostream>

#include <vector>

#include "figure.h"

class Board

{

private:

std::vector<std::vector<int> > Matrix;

const int width =16;

const int height = 20;

public:

Board();

int getWidth();

int getHeight();

void refresh();

void setFigure(Figure figure);//установить фигуру на поле

bool checkSolidLine(); // проверить заполненные линии

void delSolidLine(int); // удалить заполненные линии

void Draw();

int getMatrixElem(int i, int j);

};

#endif // BOARD\_H

### board.cpp

Изм.

Лист

№ докум.

Подпись

Дата

Лист

32

КР-НГТУ-09.03.02-13СБК-05

#include "board.h"

Board::Board()

{

refresh();

}

void Board::refresh()

{

Matrix= std::vector<std::vector<int>>(height);

for(int i=0;i<height;i++)

Matrix[i].resize(width);

for(int i=0;i<height;i++)

{

if(i==0||i==height-1)

{

for(int j=0;j<width;j++)

Matrix[i][j]=-1;

continue;

}

for(int j=0;j<width;j++)

if(j==0||j==width-1)

Matrix[i][j]=-1;

else

Matrix[i][j]=0;

}

}

int Board::getHeight()

{

return height;

}

int Board::getWidth()

{

return width;

}

void Board::Draw()

{

for(int i=0;i<height;i++)

{

for(int j=0;j<width;j++)

{

std::cout<<Matrix[i][j];

}

std::cout<<std::endl;

}

}

void Board::setFigure(Figure figure)

Изм.

Лист

№ докум.

Подпись

Дата

Лист

33

КР-НГТУ-09.03.02-13СБК-05

{

for(int i=0;i<figure.getSize();i++)

for(int j=0;j<figure.getSize();j++)

{

if(figure.getFigureMatrixElem(i,j)==0) continue;

Matrix[figure.getY()+i][figure.getX()+j]=figure.getFigureMatrixElem(i,j);

}

}

bool Board::checkSolidLine()

{

int i, j;

for(i=height-2;i>0;i--)

{

for( j=width-2;j>0;j--)

if(Matrix[i][j]==0) break;

if(j==0){ delSolidLine(i); i++;}

}

}

void Board::delSolidLine(int i)

{

for(int j=1;j<width-1;j++)

Matrix[i][j]=0;

int buf;

for(int y=i-1;y>0;y--)

for(int x=1;x<width-1;x++)

{

buf=Matrix[y][x];

Matrix[y+1][x]=buf;

}

}

int Board::getMatrixElem(int i, int j)

{

return Matrix[i][j];

}

### figureview.h

#ifndef FIGUREVIEW\_H

#define FIGUREVIEW\_H

#include "figure.h"

#include <QWidget>

#include <QPainter>

class FigureView

{

protected:

Figure \*figure;

public:

FigureView();

virtual void draw()=0;

};

class FigureViewQT:public FigureView,public QPainter

{

private:

static const int BLOCK\_SIZE = 20;

QPainter\* painter;

public:

FigureViewQT(Figure \*\_figure, QWidget \*\_widget);

~FigureViewQT();

void draw();

};

#endif // FIGUREVIEW\_H

### figureview.cpp

Изм.

Лист

№ докум.

Подпись

Дата

Лист

34

КР-НГТУ-09.03.02-13СБК-05

#include "figureview.h"

#include <QPainter>

FigureView::FigureView(){}

FigureViewQT::FigureViewQT(Figure \*\_figure, QWidget \*\_widget)

{

figure=\_figure;

painter=new QPainter(\_widget);

}

Изм.

Лист

№ докум.

Подпись

Дата

Лист

35

КР-НГТУ-09.03.02-13СБК-05

void FigureViewQT::draw()

{

if(!figure) return;

static const std::vector< QColor > COLOR\_TABLE = {

Qt::white,

Qt::yellow,

Qt::green,

Qt::red,

Qt::cyan,

Qt::magenta,

Qt::lightGray

};

for( int i = 0; i <figure->getSize() ; i++ )

{

for( int j = 0; j < figure->getSize(); j++ )

{

int type=figure->getBlockType(i,j);

if (type==0 || type == -1) continue;

if( type <= 0 || COLOR\_TABLE.size() < type )

{

continue;

}

painter->drawRect((figure->getX()+j)\*BLOCK\_SIZE, (figure->getY()+i)\*BLOCK\_SIZE, BLOCK\_SIZE, BLOCK\_SIZE);

painter->fillRect((figure->getX()+j)\*BLOCK\_SIZE+1, (figure->getY()+i)\*BLOCK\_SIZE+1, BLOCK\_SIZE-1, BLOCK\_SIZE-1, COLOR\_TABLE[ type - 1 ] );

}

}

}