

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

**ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ**

**ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ  
«ДОНСКОЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»**

**(ДГТУ)**

Факультет \_\_\_\_\_\_\_\_\_\_\_\_\_Информатика и вычислительная техника\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(наименование факультета)

Кафедра Программное обеспечение вычислительной техники и автоматизированных систем

(наименование кафедры)

|  |  |  |
| --- | --- | --- |
|  | | |
| Зав. кафедрой | | «ПОВТиАС» |
| \_\_\_\_\_\_\_\_\_\_\_\_ | | В.В. Долгов |
| (подпись) | | (И.О.Ф.) |
| «\_\_\_» | \_\_\_\_\_\_\_\_\_\_\_\_\_ 2019г. | |

**ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

к курсовому проекту (работе) по дисциплине (модулю):

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Объектно-ориентированное программирование\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(наименование учебной дисциплины (модуля))

на тему \_\_\_\_\_\_\_\_\_\_\_\_\_\_ Программная реализация игры “Крестики-нолики” \_\_\_\_\_\_\_\_\_\_\_\_

Автор проекта (работы) \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_ И. А. Стафеев \_ \_\_

подпись И.О.Ф.

Направление/специальность, профиль/специализация:

\_09.03.04\_ \_ Программная инженерия

код направления наименование направления (специальности)

Обозначение курсового проекта (работы) \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Группа \_ВПР31\_

Руководитель проекта \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ доцент, Б.В.Габрельян

подпись (должность, И.О.Ф.)

Проект (работа) защищен (а) \_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_

дата оценка подпись

Ростов-на-Дону

2019



МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

**ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ**

**ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ  
«ДОНСКОЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»**

**(ДГТУ)**

Факультет \_\_\_\_\_\_\_\_\_ Информатика и вычислительная техника \_\_\_ \_\_\_

(наименование факультета)

Кафедра Программное обеспечение вычислительной техники и автоматизированных систем (наименование кафедры)

|  |  |  |
| --- | --- | --- |
|  | | |
| Зав. кафедрой | | «ПОВТиАС» |
| \_\_\_\_\_\_\_\_\_\_\_\_ | | В.В. Долгов |
| (подпись) | | И.О.Ф. |
| «\_\_\_» | \_\_\_\_\_\_\_\_\_\_\_\_\_ 2019г. | |

**ЗАДАНИЕ**

на курсовой проект (работу)

Студент \_\_\_\_\_\_Стафеев И. А.\_\_\_\_\_\_\_ Код 09.03.04.130000.000 ПЗ Группа \_ВПР31\_

Тема \_\_\_\_\_\_ \_\_\_ «Программная реализация игры “Крестики-нолики”»\_\_\_\_\_\_\_\_\_\_

Срок представления проекта (работы) к защите «28»\_декабря 2019 г.

Исходные данные для курсового проекта (работы)

Задание, данное для курсовой работы: Методы и способы объектно-ориентированная реализация игры крестики-нолики .

|  |  |  |  |
| --- | --- | --- | --- |
| Содержание пояснительной записки | | | |
| ВВЕДЕНИЕ: | | | |
| В разделе «Введение» представлена информация, о понятии видео игры и рассмотрен краткий исторический ракурс их создания. | | | |
| Разделы основной части:  1. В разделе «Анализ задания» определяться правила игровой модели игры «Крестики-нолики». | | | |
| 2. В разделе «Теоретическом обзор» определено количество и взаимодействия классов TicTacToe и Cells, которые необходимы для реализации.  3. В разделе «Алгоритмическое конструирование» даны определения вспомогательных классов из библиотеки SFML, которые задействованы в реализации игры «Крестики-нолики».  4. В разделе «Программная реализация» описан код классов Cells и TicTacToe.  5. В разделе «Результаты работы программного средства» приведены примеры игрового процесса. | | | |
| ЗАКЛЮЧЕНИЕ: | | | |
| В «Заключении» подведены итоги курсовой работы с описанием разработанной программы в объектно-ориентированной парадигме.  Перечень графического материала   1. Скриншот 1 \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 2. Скриншот 2 \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 3. UML – диаграмма классов программы\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | | | |
|  | | | |
|  | | | |
| Руководитель проекта (работы) | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  подпись, дата | Б. В.Габрельян \_\_\_  И.О.Ф. |
|  |  |  |
| Задание принял к исполнению | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  подпись, дата | \_И. А. Стафеев \_\_\_\_  И.О.Ф. |

сОДЕРЖАНИЕ

Изм.

Лист

№ докум.

Подпись

Дата

Лист

4

02.03.03.130000.000 ПЗ

Разраб.

Стафеев И.А.

Руководит

Габрельян Б.В.

Проверил

Габрельян Б.В.

Утверд.

Долгов В.В.

Программная реализация игры “Тетрис”

Лит.

Листов

20

ДГТУ

кафедра ПОВТиАС

ВВЕДЕНИЕ 5

1 Анализ задания 6

1.1 Постановка задачи 6

1.2 Выбор языка программирования 6

2 Теоретический обзор 7

2.1 Связь между классами 7

3 Алгоритмическое конструирование 8

3.1 Класс RenderWindow 8

3.2 Класс Event 8

3.3 Класс Clock 8

3.4 Классы Sprite и Texture 8

3.5 Класс Vector2i 8

4 Программная реализация 9

4.1 Класс TicTacToe 10

4.2 Класс Cells 13

5 Реализация работы прогарммного средства 14

ЗАКЛЮЧЕНИЕ 15

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 16

Приложение А UML-диаграмма классов 17

Приложение Б Исходный код программного средства 18

**ВВЕДЕНИЕ**

Изм.

Лист

№ докум.

Подпись

Дата

Лист

5

02.03.03.130000.000 ПЗ

После создания компьютера появились компьютерные игры. Первые игры были выпущены в 1940-1950 годах в США. Первая в мире игра «Крестики нолики», работающая на ламповой вычислительной машине, была придумана студентом Стэндфордского университета.

Наиболее приближенной к современным играм оказалась созданная в 1947 году игра «Ракетный симулятор», изобретенная американцами Томасом Голдсмитом и Эст Рей Манном. Именно от этого устройства произошли экраны радаров, использовавшихся во Второй мировой войне.

Сейчас, в 21 веке, игры являются частью обычной жизни. Они представляют собой развлечение, попытку уйти из реального мира, насладиться плодами воображения создателей. И именно в наше время индустрия игр растет в невероятных проекциях.

Крестики-нолики — логическая игра между двумя противниками на квадратном поле 3 на 3 клетки или большего размера (вплоть до «бесконечного поля»). Один из игроков играет «крестиками», второй — «ноликами».[2]

Игроки по очереди ставят на свободные поля 3х3 знаки (один всегда крестики, другой всегда нолики). Первый, выстроивший в ряд 3 свои фигуры по вертикали, горизонтали или диагонали, выигрывает. Первый ход делает игрок, ставящий крестики.

В данной работе рассмотрена компьютерная версия игры «Крестики-нолики» с помощью объектно-ориентированного подхода.

1. **Анализ задания**

Изм.

Лист

№ докум.

Подпись

Дата

Лист

6

02.03.03.130000.000 ПЗ

В данном разделе определяется правила игровой модели процесса. То есть что именно должно быть реализовано и как это должно выглядеть.

* 1. **Постановка задачи**

Написать программу с двумерной графикой, которая позволяет пользователю играть в игру «Крестики-нолики».

Программа начинается с пустого игрового поля размером 3х3. Пользователь должен кликнуть левой клавишей мыши в клетку поля, куда хочет поставить крестик или нолик, но по традиции первыми ходят крестики, затем происходит ход нолика соответственно, и так происходит чередование, пока игра не будет закончена.

Цель игры – выстроить в ряд 3 свои фигуры по вертикали, горизонтали или диагонали.

Также в данной игре присутствует условие ничьи, если по истечении свободных клеток поля, никто так и не смог достигнуть условия победы.

* 1. **Выбор языка программирования**

Для выполнения поставленной задачи я воспользуюсь возможностями языка C++[1].

C++ - объектно-ориентированный язык программирования, позволяющий реализовать широкий спектр задач[1].

Для реализации графической составляющей поставленной задачи я использовал библиотеку SFML.

SFML - простая и быстрая кроссплатформенная мультимедийная библиотека, написанная на C++. Представляет собой объектно-ориентированный аналог SDL[4].

1. **Теоретический обзор**

В начале работы над задачей необходимо принять решения о количестве классов. В моей работе будет два класса.

Изм.

Лист

№ докум.

Подпись

Дата

Лист

7

02.03.03.130000.000 ПЗ

За вывода на экран нашей «графики» будет отвечать класс «TicTacToe». Он будет содержать массив, представляющий собой игровое поле. Также сделаем метод, который будет содержать в себе всю основную логику и геймплей(игровой процесс) игры.

Вторым будет класс «Cells», который будет отвечать за представление клетки поля и ее функционала на игровом поле. Помимо текстуры и спрайта клетки, данный класс содержит методы, которые свойственны поведению клетки, такие как: вывод пустой клетки, вывод клетки с крестиком и вывод клетки с ноликом.

**2.1 Связь между классами**

На старте игры задаются начальные параметры игры для классов «TicTacToe» и «Cells».

Класс «Cells» агрегирован с классом «TicTacToe». Поэтому взаимодействия между ними заключаются в том, что мы можем связать отдельные клетки с игровым полем и задать логику поведения клеток.

1. **Алгоритмическое конструирование**

Библиотека SFML позволяет нам использовать следующие заголовочные файлы: «Graphics.hpp» и «Vevtor2.hpp».

Приведенные заголовочные файлы содержат нужные для реализации поставленной задачи классы такие как RenderWindow, Event, Clock, Textutre, Sprite, Vector2i. Все эти классы находятся в пространстве имен sf[4].

Изм.

Лист

№ докум.

Подпись

Дата

Лист

8

02.03.03.130000.000 ПЗ

Изм.

Лист

№ докум.

Подпись

Дата

Лист

**3.1 Класс RenderWindow**

RenderWinodw – это класс, представляющий главное окно реализуемой игры. С помощью него мы можем создать окно, задать его размер и название.

* 1. **Класс Event**

Event – это класс, позволяющий обрабатывать события. Например, проверять нажатие клавиши или статут окна(закрыто/открыто)

* 1. **Класс Clock**

Clock – это класс, позволяющий нам измерить прошедшее время. Этот класс выполняет две функции: возвращает время, прошедшее с запуска часов, и перезапускает часы.

* 1. **Классы Sprite и Texture**

Класс Texture представляет собой изображение, которое наносят на 2D объект. Сам же объект, с уже нанесенной текстурой, является результатом работы класса Sprite.

* 1. **Класс Vector2i**

Класс Vector2i представляет собой пару целых значений. Его часто используют для хранения координат игрового объекта.

**4 Программная реализация**

Изм.

Лист

№ докум.

Подпись

Дата

Лист

9

02.03.03.130000.000 ПЗ

Для выполнения работы был выбран язык программирования C++ в среде разработки Visual Studio. Достоинствами выбранной среды являются: редактор исходного кода, отладчик кода и т.д. Visual Studio также позволяет создавать и подключать сторонние дополнения(плагины) и библиотеки для расширения функциональности.

**4.1 Класс TicTacToe**

Класс TicTacToe отвечает за основную логику игры, а также за игровой процесс. Данный класс содержит поля:

сell – объект класса Cell, для возможности представления на игровом поле клеток.

field – целочисленный двумерный массив, представляющий собой игровое поле.

coord – объект класса Vector2i, отвечающий за координаты клетки на игровом поле.

turn – булева переменная, отвечающая за ход в процессе игры.

event – объект класса Event, отвечающий за события.

Конструктор класса TicTacToe инициализирует поле field числами начиная с 2 и заканчивая 10(чтобы различать состояние конкретных клеток игрового поля). Это говорит нам о том, что игровое поле пустое. Также задаем начало первого хода игры булевой переменной turn и инициализируем Vector2i, которые отвечает за начальные координаты клетки.[3]

TicTacToe::TicTacToe()

{

turn = 0;

coord.x = 0;

coord.y = 0;

int m = 2;

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

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

{

field[i][j] = m;

m += 1;

}

}

Класс TicTacToe содержит метод startGame(). Этот метод содержит в себе основную игровую логику, а также занимается отрисовкой.

while (window.isOpen()) {

Изм.

Лист

№ докум.

Подпись

Дата

Лист

10

02.03.03.130000.000 ПЗ

while (window.pollEvent(event)) {

if (event.type == Event::Closed)

window.close();

if (event.type == Event::MouseButtonPressed)

if (event.key.code == Mouse::Left)

coord = Mouse::getPosition(window);

}

if (!turn) {

if (coord.x != 0 && coord.y != 0 && field[coord.y / 100][coord.x / 100] != 1

&& field[coord.y / 100][coord.x / 100] != -1) {

field[coord.y / 100][coord.x / 100] = 1;

turn = !turn;

}

}

else {// O

if (coord.x != 0 && coord.y != 0 && field[coord.y / 100][coord.x / 100] != 1

&& field[coord.y / 100][coord.x / 100] != -1) {

field[coord.y / 100][coord.x / 100] = -1;

turn = !turn;

}

}

for (int i = 0; i < 3; i++) {

for (int j = 0; j < 3; j++){

if (field[i][j] == 1)

cell.printX(i, j, &window);

else if (field[i][j] == -1)

cell.printO(i, j, &window);

else

cell.printCells(i, j, &window);

}

}

isWin(&window);

window.display();

}

}

Также данный класс содержит метод isWin(), который отвечает за победу в нашей игре.

Изм.

Лист

№ докум.

Подпись

Дата

Лист

11

02.03.03.130000.000 ПЗ

void TicTacToe::isWin(RenderWindow\* window) {

if (field[0][0] == field[0][1] && field[0][1] == field[0][2])

stopGame(&event, window, 0, 50, 300, 50);

if (field[1][0] == field[1][1] && field[1][1] == field[1][2])

stopGame(&event, window, 0, 150, 300, 150);

if (field[2][0] == field[2][1] && field[2][1] == field[2][2])

stopGame(&event, window, 0, 250, 300, 250);

if (field[0][0] == field[1][0] && field[1][0] == field[2][0])

stopGame(&event, window, 50, 0, 50, 300);

if (field[0][1] == field[1][1] && field[1][1] == field[2][1])

stopGame(&event, window, 150, 0, 150, 300);

if (field[0][2] == field[1][2] && field[1][2] == field[2][2])

stopGame(&event, window, 250, 0, 250, 300);

if (field[0][0] == field[1][1] && field[1][1] == field[2][2])

stopGame(&event, window, 0, 0, 300, 300);

if (field[0][2] == field[1][1] && field[1][1] == field[2][0])

stopGame(&event, window, 300, 0, 0, 300);

if (field[0][0] != 2 && field[0][1] != 3 && field[0][2] != 4 &&

field[1][0] != 5 && field[1][1] != 6 && field[1][2] != 7 &&

field[2][0] != 8 && field[2][1] != 9 && field[2][2] != 10)

stopGame(&event, window, 0, 0, 0, 0);

return;

}

И еще один метод – stopGame(). Он отвечает за остановку игры в случае победы или ничьей.

Изм.

Лист

№ докум.

Подпись

Дата

Лист

12

02.03.03.130000.000 ПЗ

void TicTacToe::stopGame(Event\* event, RenderWindow\* window, int x1, int y1, int x2, int y2)

{

Clock clock;

float time = 0;

Vertex line[] =

{

Vertex(Vector2f(x1, y1)),

Vertex(Vector2f(x2, y2))

};

line->color = Color::Blue;

window->draw(line, 2, Lines);

window->display();

while (time != 2)

{

time = clock.getElapsedTime().asSeconds();

time += time;

}

window->close();

}

**4.2 Класс Cells**

Изм.

Лист

№ докум.

Подпись

Дата

Лист

13

02.03.03.130000.000 ПЗ

Класс Cells отвечает за представление клетки игрового поля и ее функционала в нем[3]. Данный класс содержит поля:

texture – объект класса Texture, для возможности загрузить и хранить текстуру, чтобы наложить ее на спрайт.

sprite – объект класса Sprite, представляющий игровой объект и, позволяющий наложить на него текстуру.

Конструктов класса Cells инициализирует все поля, а также загружает и накладывает текстуру на спрайт клетки.

Метод printCells рисует пустую клетку в определенных координатах, координаты получаются исходя из переданных значений массива игрового поля.

void Cells::printCells(int i, int j, RenderWindow\* window) {

sprite.setPosition(j \* 100, i \* 100);

window->draw(sprite);

}

Метод printX рисует клетку с крестиком внутри в определенных координатах, координаты получаются исходя из переданных значений массива игрового поля.

void Cells::printX(int i, int j, RenderWindow\* window) {

Texture textureX;

textureX.loadFromFile("C:\\Users\\ursum\\Desktop\\TicTacToe\\Images\\X.png");

Sprite spriteX(textureX);

spriteX.setPosition(j \* 100, i \* 100);

window->draw(spriteX);

}

Метод printO рисует клетку с ноликом внутри в определенных координатах, координаты получаются исходя из переданных значений массива игрового поля.

void Cells::printO(int i, int j, RenderWindow\* window){

Texture textureO;

textureO.loadFromFile("C:\\Users\\ursum\\Desktop\\TicTacToe\\Images\\O.png");

Sprite spriteO(textureO);

spriteO.setPosition(j \* 100, i \* 100);

window->draw(spriteO);

}

**5. Результаты работы программного средства**

Изм.

Лист

№ докум.

Подпись

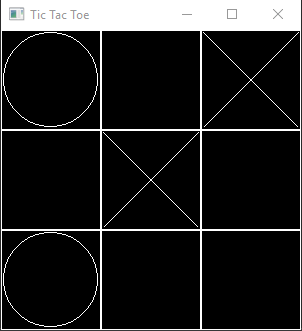
Дата

Лист

14

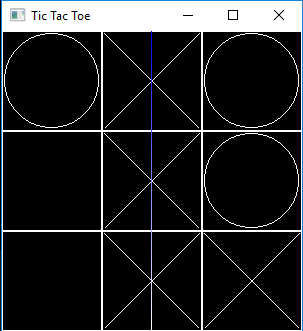
02.03.03.130000.000 ПЗ

Представлен пример игрового процесса(геймплея) и игровой модели.



Скриншот 1

Условие победы :



Скриншот 2

**ЗАКЛЮЧЕНИЕ**

Изм.

Лист

№ докум.

Подпись

Дата

Лист

15

02.03.03.130000.000 ПЗ

В курсовой работе была выполнена задача написать объектно-ориентированную реализацию игры «Крестики-нолики». Данное программное средство реализовано на языке C++ с использованием возможностей библиотеки SFML, где присутствует игровое поле, пустые клетки или клетки с фигурками. Фигурки в клетках появляются по клику левой клавиши мыши на пустые клетки и в зависимости оттого, чей на данный момент ход.

В приложении А находится UML-диаграмма программы, которая описывает связь классов в объектно-ориентированной парадигме. Также подробно описаны поля и методы каждого класса. Фрагменты исходного кода приведены в приложении Б в виде листинг Б.1 – Исходный код класса TicTacToe, листинг Б.2 – Исходный код класса Cells.

**СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ**

Изм.

Лист

№ докум.

Подпись

Дата

Лист

16

02.03.03.130000.000 ПЗ

1. С. Прата “Язык программирования C++. Лекции и упражнения”

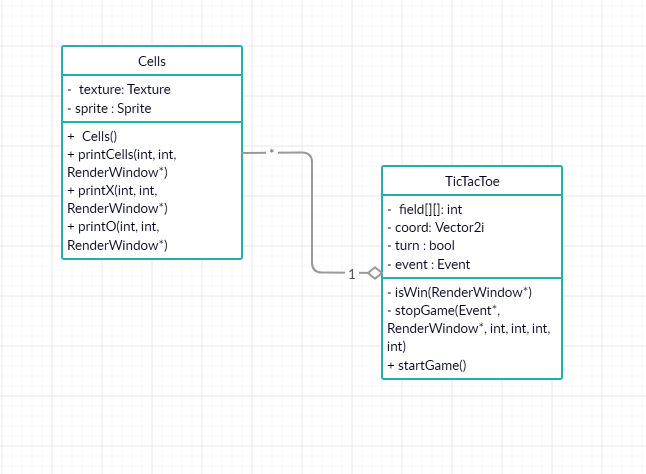
2.<https://ru.wikipedia.org/wiki/%D0%9A%D1%80%D0%B5%D1%81%D1%82%D0%B8%D0%BA%D0%B8-%D0%BD%D0%BE%D0%BB%D0%B8%D0%BA%D0%B8>

3. http://javilop.com/gamedev/

4. https://www.sfml-dev.org/

**ПРИЛОЖЕНИЯ А**

**UML-диаграмма классов**



Изм.

Лист

№ докум.

Подпись

Дата

Лист

17

02.03.03.130000.000 ПЗ

UML-диаграмма классов программы

**ПРИЛОЖЕНИЯ Б**

Изм.

Лист

№ докум.

Подпись

Дата

Лист

18

02.03.03.130000.000 ПЗ

**Исходный код программного средства**

Листинг Б.1 – Исходный код класса TicTacToe

TicTacToe::TicTacToe() {

turn = 0;

coord.x = 0;

coord.y = 0;

int m = 2;

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

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

{

field[i][j] = m;

m += 1;

}

}

void TicTacToe::startGame() {

RenderWindow window(VideoMode(300, 300), "Tic Tac Toe");

while (window.isOpen())

{

while (window.pollEvent(event))

{

if (event.type == Event::Closed)

window.close();

if (event.type == Event::MouseButtonPressed)

if (event.key.code == Mouse::Left)

coord = Mouse::getPosition(window);

}

if (!turn)

{// X

if (coord.x != 0 && coord.y != 0 && field[coord.y / 100][coord.x / 100] != 1

&& field[coord.y / 100][coord.x / 100] != -1) {

field[coord.y / 100][coord.x / 100] = 1;

turn = !turn;

}

}

else

{// O

if (coord.x != 0 && coord.y != 0 && field[coord.y / 100][coord.x / 100] != 1

&& field[coord.y / 100][coord.x / 100] != -1)

{

field[coord.y / 100][coord.x / 100] = -1;

turn = !turn;

}

}

for (int i = 0; i < 3; i++) {

for (int j = 0; j < 3; j++){

if (field[i][j] == 1)

Изм.

Лист

№ докум.

Подпись

Дата

Лист

19

02.03.03.130000.000 ПЗ

cell.printX(i, j, &window);

else if (field[i][j] == -1)

cell.printO(i, j, &window);

else

cell.printCells(i, j, &window);

}

}

isWin(&window);

window.display();

}

}

void TicTacToe::isWin(RenderWindow\* window)

{

if (field[0][0] == field[0][1] && field[0][1] == field[0][2])

stopGame(&event, window, 0, 50, 300, 50);

if (field[1][0] == field[1][1] && field[1][1] == field[1][2])

stopGame(&event, window, 0, 150, 300, 150);

if (field[2][0] == field[2][1] && field[2][1] == field[2][2])

stopGame(&event, window, 0, 250, 300, 250);

if (field[0][0] == field[1][0] && field[1][0] == field[2][0])

stopGame(&event, window, 50, 0, 50, 300);

if (field[0][1] == field[1][1] && field[1][1] == field[2][1])

stopGame(&event, window, 150, 0, 150, 300);

if (field[0][2] == field[1][2] && field[1][2] == field[2][2])

stopGame(&event, window, 250, 0, 250, 300);

if (field[0][0] == field[1][1] && field[1][1] == field[2][2])

stopGame(&event, window, 0, 0, 300, 300);

if (field[0][2] == field[1][1] && field[1][1] == field[2][0])

stopGame(&event, window, 300, 0, 0, 300);

if (field[0][0] != 2 && field[0][1] != 3 && field[0][2] != 4 &&

field[1][0] != 5 && field[1][1] != 6 && field[1][2] != 7 &&

field[2][0] != 8 && field[2][1] != 9 && field[2][2] != 10)

stopGame(&event, window, 0, 0, 0, 0);

return;

}

void TicTacToe::stopGame(Event\* event, RenderWindow\* window, int x1, int y1, int x2, int y2)

{

Clock clock;

float time = 0;

Vertex line[] =

{

Vertex(Vector2f(x1, y1)),

Vertex(Vector2f(x2, y2))

};

line->color = Color::Blue;

window->draw(line, 2, Lines);

window->display();

while (time != 100)

{

time = clock.getElapsedTime().asSeconds();

time += time;

}

window->close();

}

Листинг Б.2 – Исходный код класса Cells

#include "Cells.h"

Cells::Cells()

{

texture.loadFromFile("C:\\Users\\ursum\\Desktop\\TicTacToe\\Images\\Void.png");

sprite.setTexture(texture);

sprite.setTextureRect(IntRect(0, 0, 100, 100));

}

void Cells::printCells(int i, int j, RenderWindow\* window)

{

sprite.setPosition(j \* 100, i \* 100);

window->draw(sprite);

}

void Cells::printX(int i, int j, RenderWindow\* window)

{

Texture textureX;

textureX.loadFromFile("C:\\Users\\ursum\\Desktop\\TicTacToe\\Images\\X.png");

Sprite spriteX(textureX);

spriteX.setPosition(j \* 100, i \* 100);

window->draw(spriteX);

}

void Cells::printO(int i, int j, RenderWindow\* window)

{

Texture textureO;

textureO.loadFromFile("C:\\Users\\ursum\\Desktop\\TicTacToe\\Images\\O.png");

Sprite spriteO(textureO);

spriteO.setPosition(j \* 100, i \* 100);

window->draw(spriteO);

}

Изм.

Лист

№ докум.

Подпись

Дата

Лист

20

02.03.03.130000.000 ПЗ