СОДЕРЖАНИЕ

[ВВЕДЕНИЕ 4](#__RefHeading___Toc1411_3945178218)

[1 АНАЛИЗ ЗАДАНИЯ И ПОСТАНОВКА ЗАДАЧ 5](#__RefHeading___Toc1413_3945178218)

[2 ПРОЕКТИРОВАНИЕ ПРОГРАММЫ 7](#__RefHeading___Toc1415_3945178218)

[3 Реализация программы 13](#__RefHeading___Toc1417_3945178218)

[4 Тестирование программы 18](#__RefHeading___Toc1419_3945178218)

[Заключение 28](#__RefHeading___Toc1421_3945178218)

[ЛИТЕРАТУРА 29](#__RefHeading___Toc1423_3945178218)

# **ВВЕДЕНИЕ**

[[[Вот тут ты сам напишешь что-нибудь на пол листа на тему игр и своего отношения к ним]]]]]

В процессе написания курсовой работы необходимо закрепить знания, полученные при выполнении лабораторных работ, а так же, кроме закрепления, углубить эти знания, расширив свой кругозор. Так же в процессе написания может появится необходимость в изучении какой-либо новой технологии, позволяющей выполнить поставленную задачу и не выходящую за рамки данного курсового проекта. Углубление знаний в области программирование на языке C++ позволит писать более эффективные алгоритмы и использовать все преимущества этого языка программирования, а так же его стандартной библиотеки.

Так же важной темой для изучения является сама игра «Сапёр». Необходимо изучить механику этой игры, пришедшей к нам из ранних версий Windows, что бы, основываясь на полученном опыте, выполнить все поставленные задачи.

В качестве средств разработки были выбраны следующие средства:

* Visual Studio 2017 — современная и функциональная интегрированная среда разработки для комфортного программирования и дебага;
* компилятор msvc — компилятор компании Microsoft, идущий в комплекте с Visual Studio 2017;
* язык программированя C++ — быстрый и гибкий язык программирования высокого уровня, отлично подходящий для написания игр;
* OpenGL — графическая библиотека, предназначеная для рисования графических объектов средством графических адаптеров на любой платформе и любом железе (кроссплатформенность);
* GLUT — инструмент для быстрой, простой и удобной разработки тестовых приложений;
* Чай — чёрный чай с печеньками.

**1 АНАЛИЗ ЗАДАНИЯ И ПОСТАНОВКА ЗАДАЧ**

В данной курсовой работе предлагается разработать игровую программу «Сапёр». Для выполнения поставленной задачи требуются знания базовых возможностей языка C/C++, а также базовые знания и навыки использования библиотек Windows и OpenGL.

Главными задачами в проекте «Сапёр» являются:

1. Разработка главного окна приложения.
2. Подключение и инициализация параметров графической библиотеки OpenGL.
3. Загрузка данных (Текстуры, карты, сцены и т.д).
4. Реализация UI.
5. Генерация исходных данных.
6. Разработка логики оригинальной игры «Сапёр».
7. Возможность указывать размер поля.
8. Подсчёт времени игры.
9. Таблица рекордов на 10 записей.

Разрабатываемое игровое приложение должно эмитировать игровой процесс в игру «Сапёр». Это значит, что в игре должно присутствовать игровое поле, размер которого определяется либо программой, либо пользователем.

Игровое поле содержит «закрытые» ячейки. Под любой из ячеек игрового поля может содержаться мина. Пользователь должен открывать эти ячейки, избегая ячеек с минами. Что бы обозначить ячейку с миной, в рядом стоящих ячейках содержится число, обозначающее, сколько мин вокруг этой ячейки находится. Учитываются только те мины, что лежат в радиусе одной ячейки, относительно исходной. Игра считается выигранной, если пользователь откроет все ячейки, не содержащие мины. Это правило работает и в обратную сторону: игра считается проигранной, если пользователь откроет любую ячейку с миной. Так же в оригинальной игре присутствуют «флаги», которые можно ставить на любую закрытую ячейку. Таким образом помечается, что в этой ячейке может быть мина и эту ячейку нельзя открыть кликом мыши, пока на ней весит «флаг». Количество флагов ограничено количеством мин на игровом поле.

Графический интерфейс пользователя должен обладать следующими характеристиками:

* удобство использования — пользователь не должен испытывать дискомфорт при использовании игрового приложения;
* эстетичность — внешний вид интерфейса не должен вызывать у пользователя неприятные эмоции из-за своего внешнего вида;
* интуитивность — интерфейс должен быть чётко проработанным, что бы пользователь без подготовки мог сразу приступить к использованию приложения.

Пользователь должен видеть, что он может сделать, что ему доступно в данный момент. Если это главное меню — кнопки для начала игры и прочее, если это игровая сцена — игровое поле и прочие атрибуты. Пользователь, глядя на них, должен понимать, что перед ним находится и так далее.

Восстановления игрового процесса, после завершения сеанса игры необходима на тот случай, если пользователь не успел закончить игровой сеанс по каким-то сторонним причинам, которые не будут рассматриваться в рамках данного проекта. Важно лишь реализовать возможность восстановления последней игры, что бы пользователь в любой момент смог закончить свою игру. Не стоит так же забывать об автосохранении игрового процесса на случаи непредвиденного завершения игрового приложения, путём выключения компьютера или ручной остановки процесса игрового приложения.

Таблица рекордов игрового приложения должна содержать записи о десяти самых лучших игроках, которые играли в эту игру и получили лучшим результаты. Результат игры считается лучшим, если он содержит в себе наименьшее время и наименьшее количество ходов. В разрабатываемом игровом приложении «Сапёр» в первую очередь должно учитываться соотношений количества мин к размеру поля, а только потом время, потраченное на открытие всех ячеек игрового поля.

**2 ПРОЕКТИРОВАНИЕ ПРОГРАММЫ**

Почти все игровые приложения начинают своё общение с пользователем, предоставив последнему своё главное игровое меню. В главном игровом меню должна быть реализована возможность начала самой игры, восстановление предыдущей сессии, просмотр таблици лидеров, а так же механизм выхода из игры.

Что бы пользователю было понятно, что делать, что бы получить какой-то результат, нужно обратить внимание в сторону классического элемента управления — кнопку, вместо всех современные способов реализации взаимодействия с пользователем. Классические кнопки знакомы огромному числу лиц и не должны вызывать трудности при работе с ними. Что бы пользователь понял, что перед ним кнопка, она должна выделяться на фоне остального пространства, а так же изменять цвет при наведении на неё.

Решив вопрос с элементами управления перейдём к проектированию главного меню. Главное меню должно содержать четыре кнопки, с помощью которых будут осуществляться переходы в другие разделы. Каждая кнопка должна содержать короткий текст, описывающий её предназначение. Следующие кнопки должны быть в главном меню игрового приложения:

1. «Новая игра» - для начала новой игры;
2. «Продолжить игру» - для восстановления последнего игрового сеанса;
3. «Лидеры» - для получения списка самых лучших игроков;
4. «Выход» - для завершения игрового процесса.

Примерный внешний вид главного меню игрового приложение представлен на рисунке 2.1.

Игровая сцена должна содержать в себе игровое поле с расположенными на нём игровыми фишками, время, прошедшее с момента начала игры, счётчик ходов, показывающий, сколько ходов сделал пользователь, а так же кнопки управления для внедрение дополнительных возможностей, таких как начало новой игры и выход в главное меню. Примерный внешний вид игрового поля представлен на рисунке 2.2.

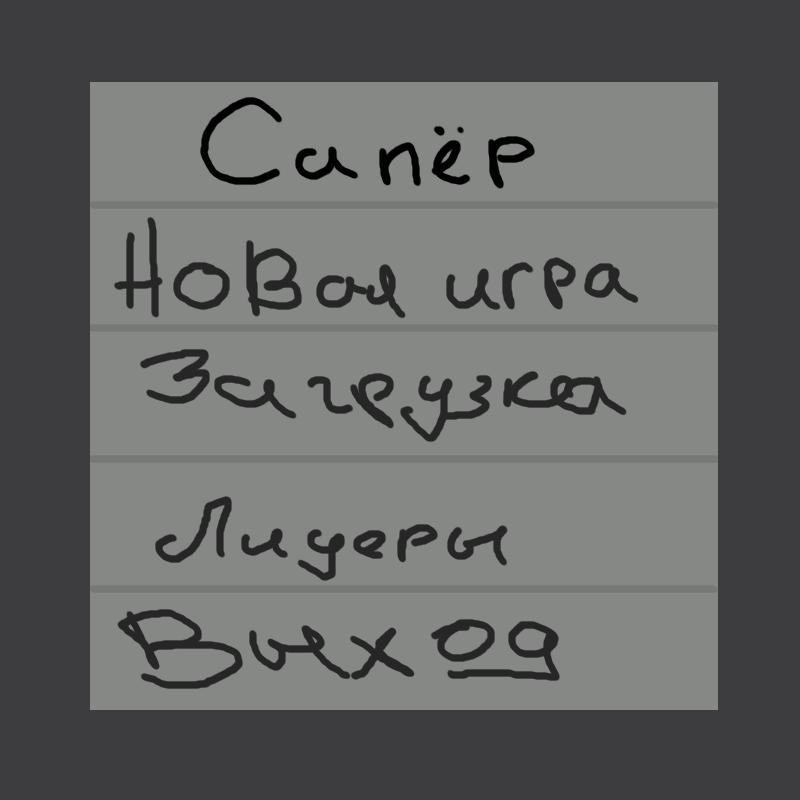
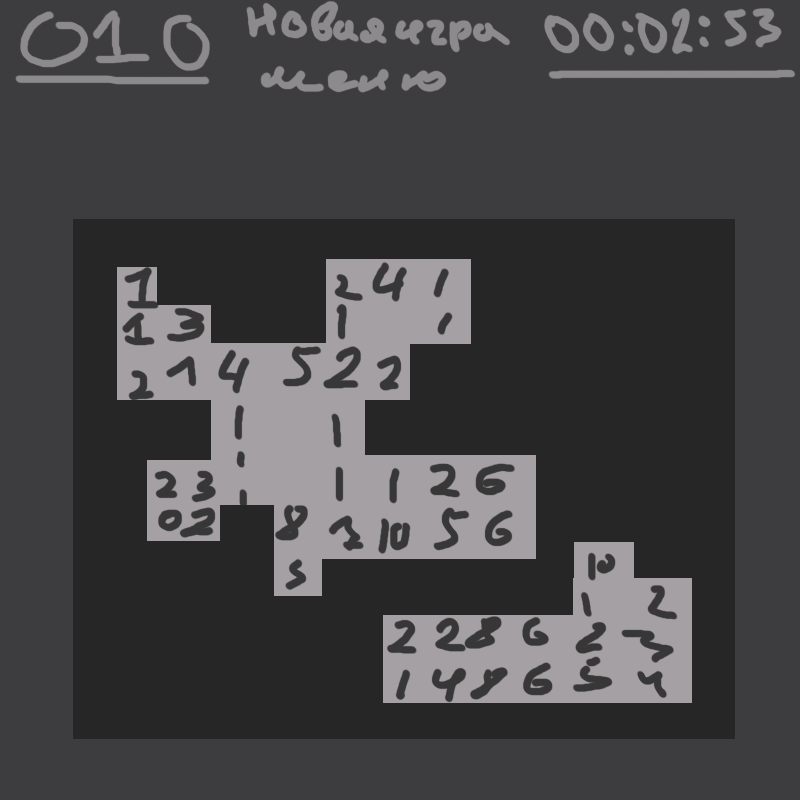
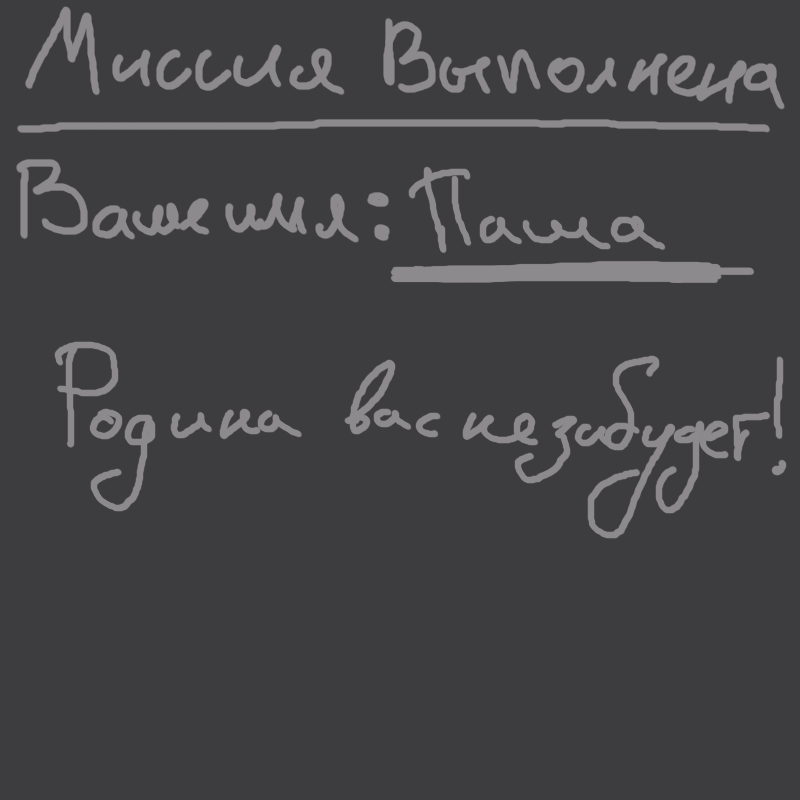


Рисунок 2.1 - Главное меню

После того, как пользователь решил головоломку, он должен быть осведомлён о своей победе. Кроме того пользователю надо ввести данные о том, как его зовут. Всё это можно объединить в одну сцену, в которой пользователь увидит информацию о том, что он выиграл, после чего введёт своё имя. Эта сцена должна отображать что-то, что говорит о победе пользователя, иметь поле для ввода имени пользователя и кнопку подтверждения ввода. Примерный внешний вид окна «победы» представлен на рисунке 2.3.

Рисунок 2.2 - Игровое поле

После победы пользователь может захотеть ознакомиться с результатами других игроков, играющих в данную игру. Для этих целей в главном меню была предусмотрена кнопка «Лидеры», которая должна перенаправить пользователя на страницу просмотра результатов других игроков. Сама сцена должна отображать список из 10 записей, содержание которых указано в главе один, а так же кнопку, позволяющую вернуться в главное меню игрового приложения. Примерный внешний вид сцены со списком лидеров на десять записей представлен на рисунке 2.4

Рисунок 2.3 - Победа

Для реализации графической составляющей игрового приложения выбор пал на бесплатную библиотеку FreeGLUT, которая предоставляет интерфейс для создания окна с контекстом OpenGL и управления этим окном, инкапсулируя в себе всё, что связано с процессом создания окна, созданием контекста OpenGL и обработки разных событий ввода/вывода. Эта библиотека и станет основой разрабатываемого игрового приложения «Сапёр».

Рисунок 2.4 - Лидеры

Что бы не создавать огромного количества ветвлений в функциях-обработчиках событий, была выбрана модель, позволяющая обрабатывать каждую игровую сцену независимо друг от друга. Принцип работы следующий: есть какая-то структура, хранящая в себе адреса функций обработки событий рисования, клавиатуры и мыши. Один объект такой структуры представляет из себя одну сцену. Заполнение этих объектов сцен нужно реализовать в функции, которая отвечает за подготовку приложения к игре при старте. Что бы не производить постоянные установки обработчиков событий для окна GLUT, функции, указанные в объектах сцен будут вызываться прямиком из обработчиков событий, установленных по-умолчанию.

Библиотека GLUT не содержит встроенных элементов управления, таких как кнопки и текстовые поля, поэтому этот функционал будет восполняться вручную, путём создания функций, отвечающих за рисование кнопок. Для упрощения реализации каждая кнопка будет иметь всего два состояния: без курсора и с курсором. Кнопка без курсора окрашивается одним цветом, а кнопка с курсором окрашивается другим, что бы пользователь смог увидеть активность данного элемента. Кнопка считается нажатой в том случае, когда пользователь производит клик мыши, при этом курсор мыши должен находится над самой кнопкой.

Вопрос с текстовым полем сложнее, так как и принцип его работы сильно отличается от обычной кнопки. По этой причине будет реализована простая версия текстового поля, которая будет иметь возможности добавлять и удалять символы из конца строки. Такое решение позволит ускорить процесс разработки, так как не является сложным в реализации. Этому способствует наличие в библиотеки GLUT встроенных методов работы с клавиатурой, а так же заготовленные растровые шрифты, которые можно использовать для отображения текста.

Игровая механика игры основывается на открытие ячеек, содержащих либо подсказку, либо мину. Это означает, что приложению необходима реализация маханики генерации игрового поля, открытия плиток, а так же расстановки флажков, предупреждающих о наличии мин. Для расчёта удобства проверки поля на состояние «победы» необходимо реализовать дополнительную структуру с данными, которая будет хранить информацию обо всех минах. Таким образом время проверки состояния поля будет сведено к минимуму.

**3 Реализация программы**

Работа программы начинается с функции main. Именно в этой функции инициализируем GLUT и создаём окно. После успешной инициализации GLUT-а и создания окна поочерёдно вызываем функции инициализации:

1. initializationGl — функция инициализации свойств OpenGL.
2. initializationGlut — функция инициализации Glut.
3. initializationGame — функция инициализации самой игры.

На начальном этапе эти функции будут пустовать, изредка пополняясь дополнительным кодом.

Для рисования всего содержимого окна, а так же для обработки событий мыши и клавиатуры регестрируем в GLUT callback функции, которые будут отвечать за первостепенную обработку события, вызов обработчика сцены и пост обработку события:

1. mainRendererFunction — функция обработки события перерисовки окна.
2. mainKeyboardFunction — функция обработки событий нажатия клавиш.
3. mainMouseClickFunction — функция обработки событий кликов мыши.
4. mainMouseMotionFunction — функция обработки событий движения курсора.
5. mainResizeFunction — функция обработки событий изменения размера окна.

Регистрацию всех этих функций опишем в функции initializationGlut.

В функции initializationGl включим «мультисемплинг», альфа-смешивание и установим цвет, которым будет каждый раз очищаться буфер цвета.

InitializationGame — функция, которая будет отвечать за инициализацию игровых данных. На текущем этапе создадим глобальный массив объектов типа «Сцена», а в функции инициализации каждую сцену заполним соответствующими функциями. Так же создадим переменную, хранящую индекс текущей сцены в массиве и проинициализируем её в функции инициализации игры. С помощью индекса сцены нужный объект сцены будет выбираться в массиве сцен и его функция, в случае существования, будет вызываться. Такое решение избавляет от необходимости писать много разветвляющих конструкций.

После подготовки основы реализуем, непосредственно, саму игру. Начать стоит с рисования игрового поля на экране. Для этого напишем функции, отвечающие за рисование составных частей игрового поля:

* drawField;
* drawCells;
* drawHeader;
* drawFlagsLimits;
* drawTime;
* drawGameOver().

Функция drawField отвечает за рисование подложки игрового поля, а функция drawCells отвечает за рисование как скрытых, так и открытых объектов игрового поля.

Перед тем, как начать рисовать игровое поле, его надо создать. Для этого в глобальном пространстве была создана отдельная переменная, названная g\_game. Переменная g\_game — объект структуры Game, отвечающий за хранения всей игровой информации. Частью этой структуры является объект структуры, хранящий данные об игровом поле, такие как размеры игрового поля, матрица самого игрового поля, количество бомб, количество фалгов и координаты всех бомб на поле. В матрице игрового поля будет храниться информация о всех ячейках игрового поля. Число 0 — пустое поле, число -1 — бомба, а числа от нуля до восьми — это подсказки, указывающие, сколько мин находится вокруг заданной ячейки.

Вернёмся к функции drawCells. Данная функция вселишь вызывает другие функции для рисования каждой ячейки игрового поля:

* drawCellNumber;
* drawMine;
* drawCellFlag;
* drawFillCell.

Функция drawCellNumber рисует цифру ячейки в нужном месте, основываясь на индексах этой ячейки, переданных в качестве параметров. Функция drawMine используется так же, как и drawCellNumber, но вместо цифры рисуется бомба. Функция drawCellFlag рисует флажок над ячейкой, которую пользователь пометил как подозреваемую. Функция drawFillCell рисует цветной квадратик над ячейкой.

После того, как мы увидели игровое поле, его надо подготовить к игре, так как в данный момент оно является изначально пустым. Для этого напишем функцию makeNewGame. Функция makeNewGame будет отвечать за инициализацию игровой сессии, а именно за расстановку мин на игровом поле, расстановку подсказок и установку таймера. В этой функции реализуем алгоритм распределения мин по полю Сам алгоритм достаточно прост: сначала вся матрица перегоняется в массив для удобства работы, затем первые N элементов массива получают значения, соответствующие значениям бомб на поле, где N равно количеству бомб в игровой сессии, после чего все его элементы перемешиваются псевдослучайным образом. После этого массив копируется обратно в матрицу. Перемешав все бомбы по полю начинается расчёт подсказок. Процедура расчёта подсказок работает следующим образом: начинается обход полученной матрицы. Если ячейка заполнена бомбой, то все её ячейки увеличивают своё значение на один. Так же найденная бомба заносится в список бомб. Таким образом поле заполняется подсказками.

Игровое поле сгенерировано, следующий этап — открытие ячеек в это поле. Для открытия фишек напишем функцию openCell, которая будет открывать ячейки игрового поля. Функция будет принимать индексы ячейки, после чего воспользуется ими для открытия. Однако способ с открыванием одной клетки в современных играх «Сапёр» работает чуточку сложнее, чем просто открытие клетки. В игре есть клетки, которые не содержат подсказок. Это значит, что их открытие ни к чему не приведёт. Таких ячеек может быть очень много, поэтому для ускорения процесса игры добавим функцию openFreeCells, которая будет открывать все близлежащие свободные ячейки, при условии, что клик был сделан по ячейке без подсказки. Функция openFreeCells будет окрывать все ячейки без подсказок, а так же все крайние ячейки с подсказками. Так как в игре ещё присутсвуют флаги, то этот момент тоже стоит учесть и внедрить дополнительную проверку, связанную с флагами.

Функция открытия всех ячеек без подсказки работает следующим образом: создаётся очередь для ячеек, после чего в неё заносятся индексы первой нажатой ячейки. Далее начинает выполнятся цикл, в котором, на каждой итерации, извлекаются индексы первой ячейки в очереди. Если эта ячейка — ячейка с подсказкой, то все её соседние ячейки не открываются. Если ячейка — бомба, то она не открывается. Если ячейка — без подсказки, то в очередь заносятся все её соседние ячейки, после чего начинается новая итерация цикла. Цикл будет работать до тех пор, пока вся очередь не останется пустой. Исходный код функции openFreeCells представлен в листинге 3.1.

Что бы понять, какую фишку надо открыть, для начала надо реализовать функцию обработки событий движения мыши. В этой функции будут обрабатываться все перемещения мыши и регистрироваться наведения мыши на кнопку или на игровую ячейку. Пока что реализуем наведение курсора на игровую ячейку. В функции обрабатываем значения положения курсора мыши в пределах окна и переводим их к нужному нам виду, после чего все данные записываем в структуру g\_gameMouse — структуру, хранящую в себе информации о положении курсора мыши и о нажатиях кнопок мыши. После того как мы обработали событие передвижения мышии определили, что курсор находится над игровой ячейкой, надо обработать момент, связанный с нажатием кнопки мыши. Для этого напишем обработчик событий нажатия кнопок мыши и укажем в нём, что при на ведении на ячейку и нажатии над ней левой кнопки мыши эта ячейка должна открыться, если она ещё не открыта.

Листинг 3.1 — функция открытия всех свободных ячеек.

1 void openFreeCells(int startCell\_x, int startCell\_y)

2 {

3 queue<pair<int, int>> queueOfClosedCells;

4 queueOfClosedCells.push({startCell\_x, startCell\_y});

5 while(!queueOfClosedCells.empty()) {

6 auto cellCoords = queueOfClosedCells.front();

7 queueOfClosedCells.pop();

8 int x = cellCoords.first;

9 int y = cellCoords.second;

10 if (x >= 0 && x < g\_game.field.width

11 && y >= 0 && y < g\_game.field.height

12 && !g\_game.field.field[y][x].isOpened) {

13 g\_game.field.field[y][x].isOpened = true;

14 } else {

15 continue;

16 }

17 if (g\_game.field.field[y][x].isSuspect) {

18 g\_game.field.field[y][x].isSuspect = false;

19 g\_game.field.flagsAmount++;

20 }

21 if (g\_game.field.field[y][x].type != 0) {

22 continue;

23 }

24 for (int i = -1; i < 2; i++) {

25 queueOfClosedCells.push({x + i, y - 1});

26 if (i != 0) {

27 queueOfClosedCells.push({x + i, y});

28 }

29 queueOfClosedCells.push({x + i, y + 1});

30 }

31 }

32 }

Следующий этап — кнопки. Кнопки реализовать достаточно просто. Вводим перечисление ID кнопок и пишем функцию рисования кнопок — drawButton. Функция рисования кнопки будет рисовать кнопку в нужном месте и, в зависимости от её состояния, выбирать ей цвет. Состояние определяется наличием курсора над кнопкой, а это означает, что функция рисования кнопки будет использовать данные из структуры g\_gameMouse. События кликов мыши будут работать так же, как и в случае с игровыми фишками: сделал клик над кнопкой — выполнилось действие.

Из полученных функций создаём две кнопки: «новая игра» и «главное меню». Кнопка «новая игра» будет генерировать новое поле и сбрасывать игровые данные, а кнопка «главное меню» позже будет переносить игрока в главное меню игрового приложения.

Счётчик времени можно реализовать, написав одну callback функцию, которая будет вызываться каждую секунду. В GLUT предусмотрен механизм работы таймеров, однако зарегистрированный таймер уже нельзя уничтожить, поэтому в структуру игры добавим такой параметр, как gameID, что бы таймер мог вовремя остановится, когда увидит, что считает время уже не для своей игры. ID текущей игры передаётся таймеру при его создании.

Что бы убедится в победе игрока — напишем функцию checkVictory. Эта функция будет проверять игровое поле на состояние победы. В объекте игрового поля есть данные о расположении всех мин на игровом поле. Воспользовавшись этой информацией можно выполнить проверку, связанную с победой. Если все мины на игровом поле окружены открытыми ячейками, то игра считается законченной. Если игра была закончена, то функция меняет текущее игровое поле.

Игровая сцена готово, играть можно. Теперь подобным образом реализуем все остальные сцены. Однако стоит обратить внимание на сцену, появляющаяся после победы игрока. В этой сцене необходимо реализовать ввод с клавиатуры имени игрока, что бы зафиксировать его в таблице рекордов. Для этого напишем простой текстовый редактор, который будет обрабатывать только последний символ строки. После подтверждения ввода нажатием на кнопку, данные об игроке заносятся в 11 линию списка, после чего список сортируется. Таким образом мы получаем список рекордов на 10 записей.

Весь исходный код программы можно будет посмотреть в приложение.

**4 Тестирование программы**

Составим простой план тестирования готового приложения для оценки результатов работы. С планом можно ознакомиться в таблице 4.1.

Таблица 4.1 — план тестирования.

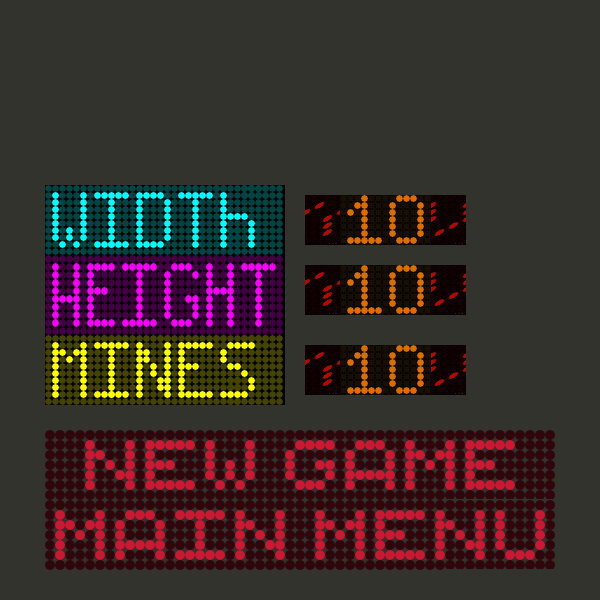
| № | Действие | Описание | Результат |
| --- | --- | --- | --- |
| 1 | Запуск программы | Проверка работоспособности программы | Окно программы с главным меню |
| 2 | Наведение на кнопку | Проверка реакции программы на наведение курсора мыши на кнопку | Кнопка должна изменить цвет |
| 3 | Нажатие на кнопку «Новая игра» | Проверка реакции программы на нажатие кнопки мыши над кнопкой | Должно открыться окно с выбором размера поля |
| 4 | Нажатие на кнопку «Пользовательский профиль» | Проверка, будет ли открыта сцена с настройкой игрового поля | Должна отобразиться сцена с настройкой игрового поля |
| 5 | Изменение параметров игры | Проверка на возможность менять размер поля и количество мин. | Размеры поля и количество мин должны меняться |
| 6 | Нажатие на кнопку «Начать игру» | Проверка на появление настроенного поля после нажатия на кнопку | Должна появиться игровая сцена и поле с заданными параметрами |
| 7 | Нажатие на не открытую ячейку левой кнопкой мыши | Проверка открытия ячеек | Ячейка должна открыться |
| 8 | Нажатие на не открытую ячейку правой кнопкой мыши | Проверка установки флагов | Должен установиться флаг |
| 9 | Проигрыш | Проверка на возможность проигрыша | Игра должна вывести сообщение о проигрыше |
| 10 | Нажатие на кнопку «Новая игра» в игре | Проверка, будет ли сгенерировано новое поле после начала новой игры | Поле должно быть перегенерировано |
| 11 | Нажатие на кнопку  «Главное меню» в игре | Проверка на выход в главное меню из игры | Текущей сценой должно стать главное меню |
| 12 | Нажатие на кнопку  «Продолжить игру» | Проверка на восстановление последней сессии | Последняя игра должна быть востановлена |
| 13 | Победа | Проверка на возможность победы | Игра должна быть проходимой |
| 14 | Нажатие «Юху» без ввода имени | Проверка на возможность не указывать своё имя после победы | Нажатие кнопки ни к чему не привело |
| 15 | Ввод имени | Проверка корректности ввода и отображения имени пользователя | Видимое имя пользователя |
| 16 | Нажатие «Юху» после ввода имени | Проверка на корректность подтверждения ввода | Должна открыться сцена «Рекорды» |
| 17 | Наличие в списке | Проверка на появление данных о победителе в пустом списке | Данные победителя должны появится |
| 18 | Нажатие на кнопку «Ок» | Проверка на переход в главное меню после нажатия | Текущей сценой должно стать главное меню |

1. Запуск программы. Запустив программу пользователь приложения должен увидеть главное меню игры. Выполнив запуск, видим окно с кнопками. Это очень похоже на главное меню. Тест пройден, результаты теста на картинке 4.1.
2. Наведение на кнопку. При наведении курсора на любую из кнопок главного меню та должна изменить цвет, сигнализируя о том, что на неё можно нажать. Наведём курсор на кнопку и увидим другой цвет. Тест пройден, результаты теста на картинке 4.2.
3. Нажатие на кнопку «новая игра». После нажатия на кнопку «Новая игра» пользователь должен увидеть сцену с выбором размера поля и количества мин. Тест пройден, результаты теста на картинке 4.3.
4. Нажатие на кнопку «Пользовательский профиль». После нажатие на кнопку «Пользовательский профиль» пользователь должен увидеть сцену с настройкой игрового поля. Тест пройден, результаты теста на картинке 4.4.

Рисунок 4.1 — Результат теста запуска программы.

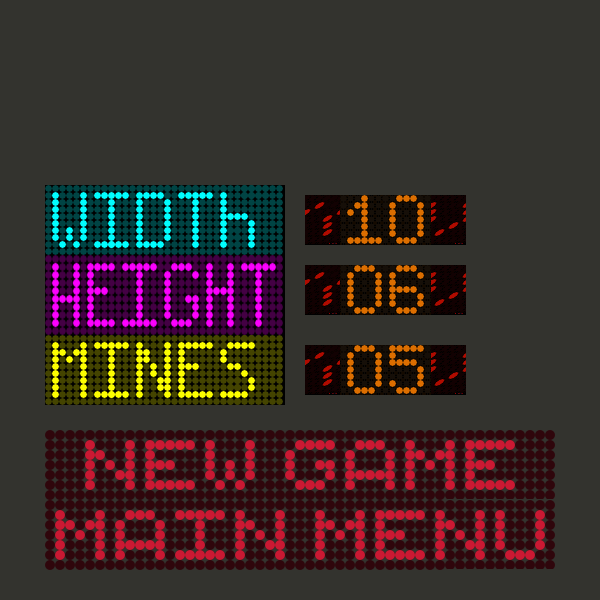
Рисунок 4.2 — Результат теста наведения на кнопку.

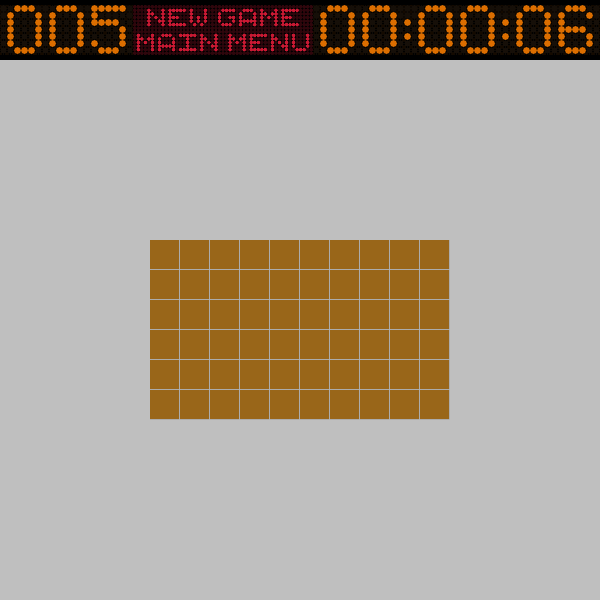
Рисунок 4.3 — Результат теста нажатия на кнопку «Новая игра».

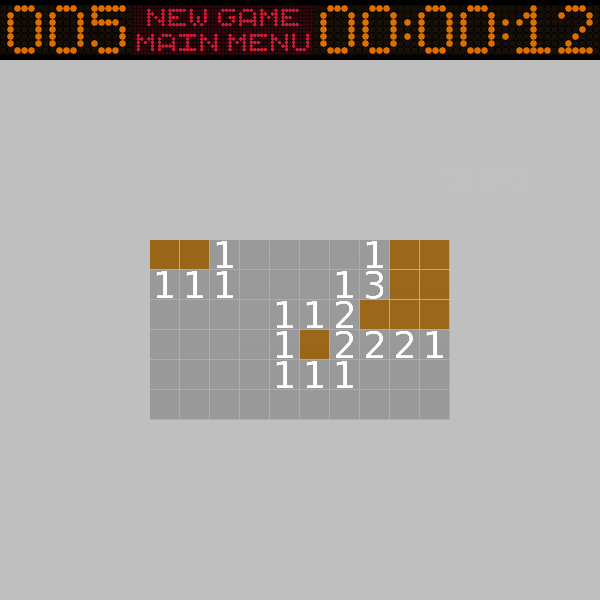
Рисунок 4.4 — Результат теста перехода к настройке игры.

1. Изменение параметров игры. Попробуем изменить параметры игры, покликав на кнопки рядом с цифрами. В результате цифры рядом с кнопками должны изменить свои значения. После нажатий на кнопки рядом с цифрами параметры игры были изменены. Тест пройден, результаты теста на картинке 4.5.
2. Нажатие на кнопку «Начать игру». Нажмём на кнопку «Начать игру». В результате должна появится игровая. После нажатия на кнопку появляется игровая сцена с полем выбранного размера. Тест пройден, результаты теста на картинке 4.6.
3. Нажатие на не открытую ячейку левой кнопкой мыши. Нажмём на одну из ячеек левой кнопкой мыши, которые не была до этого открыта. В результате эта ячейка должна открыться, а рядом с ней, по возможности, и другие ячейки. Нажимаем на выбранную ячейку и наблюдаем открытие этой ячейки. Тест пройден, результаты теста на картинке 4.7.
4. Нажатие на не открытую ячейку правой кнопкой мыши. Нажмём на одну из ячеек правой кнопкой мыши, которые не была до этого открыта. В результате эта ячейка должна пометиться флажком. Нажимаем на выбранную ячейку и наблюдаем появление флажка над этой ячейкой. Тест пройден, результаты теста на картинке 4.8.
5. Проигрыш. Попробуем найти ячейку с миной. В результате нажатия на ячейку с миной должно появится сообщение о проигрыше. Нажимаем на ячейку с миной и видим сообщение о проигрыше. Тест пройден, результаты теста на картинке 4.9.
6. Нажатие на кнопку «Новая игра» в игре. Нажав на кнопку «Новая игра», игра должна начаться заново с теми же параметрами игрового поля. После нажатия на кнопку «Новая игра» игра началась заново с уже новым полем, но с таким же размером и таким же числом мин. Тест пройден, результаты теста на картинке 4.10.
7. Нажатие на кнопку «Главное меню» в игре. Нажав на кнопку «Главное меню» пользователь должен увидеть главное меню игры. Нажимаем на кнопку «Главное меню» и видим главное меню игры. Тест пройден.
8. Нажатие на кнопку «Продолжить игру». При нажатии на кнопку «Продолжить игру» программа должна восстановить последнюю игровую сессию. Нажмём на кнопку «Продолжить игру» и видим поле, сгенерированное в одном из предыдущих кейсов. Тест пройден.
9. Победа. Открыв все ячейки без бомб должно появиться уведомление о победе с таблицей рекордов без новой записи. Выиграв игру появилась такая информация и не изменённая таблица лидеров. Тест пройден, результаты теста на картинке 4.11.
10. Нажатие «Юху» без ввода имени. При нажатии на кнопку «Юху», не указав своего имени, программа не должна реагировать. Нажмём на кнопку «Юху» не введя имя. Программа не отреагировала. Тест пройден.
11. Ввод имени. При нажатии кнопок клавиатуры в текстовом поле должны появляться вводимые символы. Вводим имя «Pasha» и видим результат на экране. Тест пройден, результаты теста на картинке 4.12.
12. Нажатие «Юху» после ввода имени. После ввода имени кнопка «Юху» должна отправлять пользователя к обновлённой таблице рекордов. Нажимаем на кнопку «Юху» и и видим таблицу рекордов. Тест пройден, результаты теста на картинке 4.13.
13. Наличие в списке. В обновлённом списке рекордов должна отображаться информация о победителе, если он один из первых, либо сильнейший. Так как это первая игра, то запись об игроке «Pasha» должна быть первой в списке. В списке запись «Pasha» на первом месте. Тест пройден.
14. Нажатие на кнопку «Ок». При нажатии на кнопку «Ок» должен произойти возврат в главное меню. Нажимаем на кнопку «Ок и попадаем в главное меню. Тест пройден.

Все тесты были успешно пройдены, что говорит о работоспособности данного игрового приложения «Сапёр».

Рисунок 4.5 — Результат теста изменения параметров игры.

Рисунок 4.6 — Результат теста начала игры.

Рисунок 4.7 — Результат теста клика ЛКМ по закрытой ячейке.

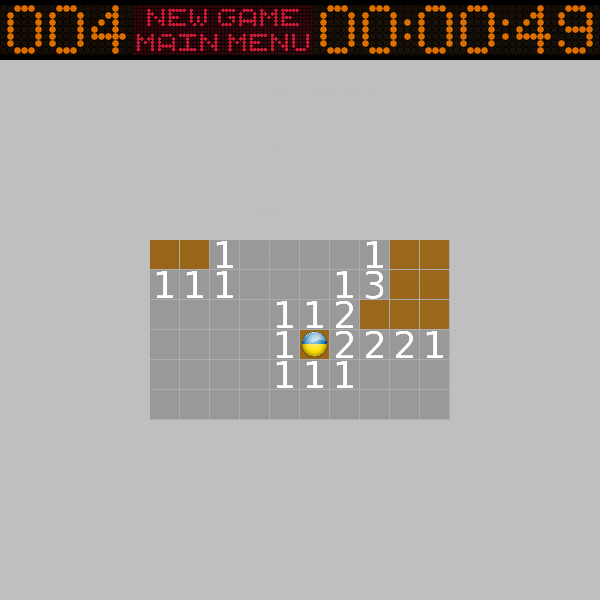
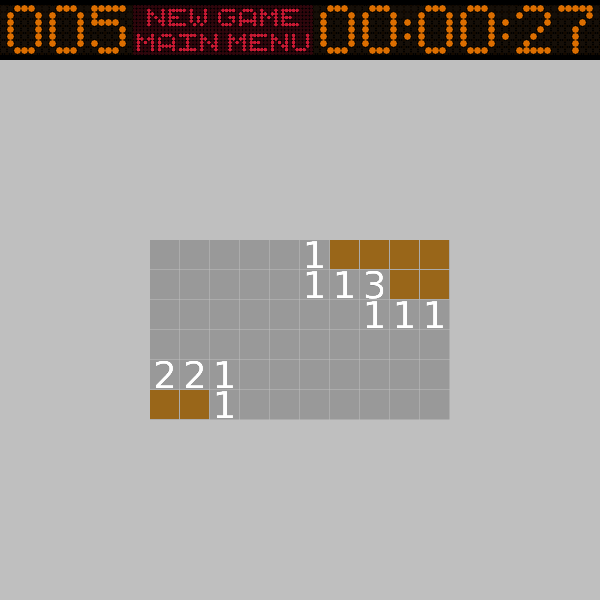
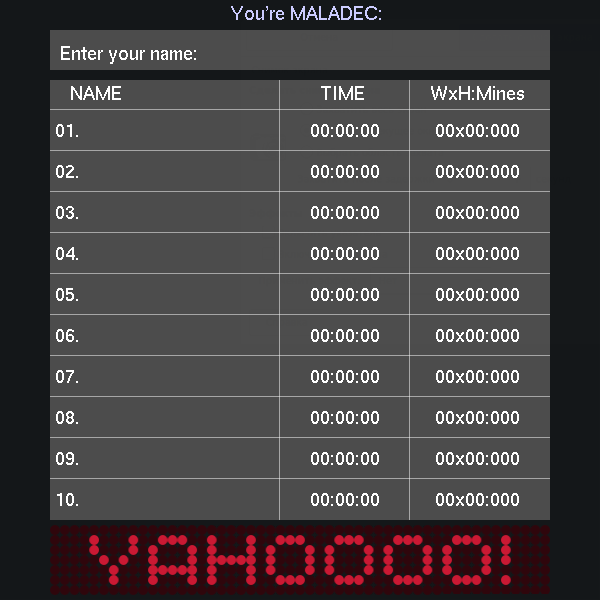
Рисунок 4.8 — Результат теста клика ПКМ по закрытой ячейке.

Рисунок 4.9 — Результат теста проигрыша.

Рисунок 4.10 — Результат теста начала новой игры с сохранением параметров.

Рисунок 4.11 — Результат теста победы.

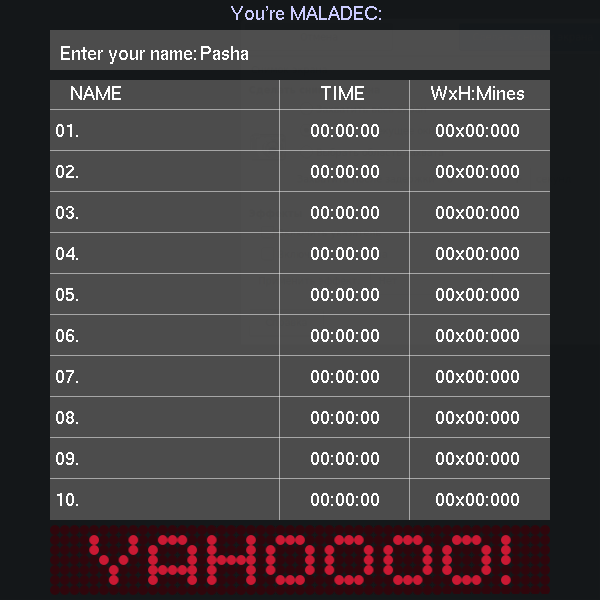
Рисунок 4.12 — Результат теста ввода имени

Рисунок 4.13 — Результат теста отображения данных в таблице.

**Заключение**

В ходе выполнения курсовой работы было написано игровое приложение «Сапёр». Данное игровое приложение должно было реализовать игровую механику классического сапёра. Так же важной задачей было написание удобного и красивого интерфейса пользователя.

В ходе разработки пришлось решить ряд задач, связанных не только с механикой самой игры, но и связанных с реализацией интерфейса пользователя, так как для разработки данного игрового приложения использовались OpenGL библиотека совместно с GLUT.

Так же был интересен момент, связанный с игровыми сценами, так такого рода задача мною решалась впервые и я считаю, что данное решение себя полностью оправдывает, так как не вносит дополнительного усложнения кода всевозможными конструкциями ветвления.

Тесты показали, что все задачи, поставленные при анализе исходных данных и проектирование выполнены полностью, а само игровое приложение является лёгким в использовании, быстрым, удобным и красивым.

**ЛИТЕРАТУРА**

1. Ричард С. Райт, мл., Бенджамин Липчак OpenGL. Суперкнига, 3-е издание.: Пер. с англ – М.: Издатель- ский дом "Вильямс", 2006. – 1040 с.
2. OpenGL: Руководство по программированию M. By, Т. Девис, Дж. Найдер, Д. Шрайнер; Пер. с анг.: Е. Васильева, Е. Эрмана. – 4-е изд,. – СПБ.: Питер, 2006 – 624 с: ил. Гмурман.