Государственное образовательное учреждение высшего

профессионального образования

«Московский Государственный Технический Университет имени

Н.Э. Баумана»

(МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ

«ИНФОРМАТИКА И СИСТЕМЫ УПРАВЛЕНИЯ»

КАФЕДРА

«ИНФОРМАЦИОННАЯ БЕЗОПАСНОСТЬ»

Руководитель проекта: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ / Мацак И. В.

(подпись, дата)

Разработчик проекта: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ / Кадыков В. Д.

(подпись, дата)



Москва 2020

КУРСОВОЙ ПРОЕКТ

Сетевая игра «Go»

Оглавление

[Цель](#__RefHeading___Toc1244_3696431472) 3

Основные определения3

[Введение](#__RefHeading___Toc1248_3696431472) 4

[Требования к проекту](#__RefHeading___Toc1250_3696431472) 4

Проектирование системы5

Выбор технологий7

Описание технических решений [1](#__RefHeading___Toc1256_3696431472)1

Заключение [1](#__RefHeading___Toc1258_3696431472)2

Список использованных источников [1](#__RefHeading___Toc1260_3696431472)2

ЦЕЛЬ

Создание сетевой игры «Go» с графическим интерфейсом на алгоритмическом языке программирования Си++ при помощи знаний и навыков, полученных на 1 курсе по специальности «Информационная безопасность». Изучение графической библиотеки SFML.

**ОСНОВНЫЕ ОПРЕДЕЛЕНИЯ**

**Алгоритмический язык программирования** - формальный язык, используемый для записи, реализации и изучения алгоритмов. В отличие от большинства языков программирования, алгоритмический язык не привязан к архитектуре компьютера, не содержит деталей, связанных с устройством машины.

**Объектно-ориентированное программирование** (ООП) - методология программирования, основанная на представлении программы в виде совокупности объектов, каждый из которых является экземпляром определенного класса, а классы образуют иерархию наследования.

**SFML** (англ. Simple and Fast Multimedia Library - простая и быстрая мультимедийная библиотека ) - свободная кроссплатформенная мультимедийная библиотека, обеспечивающая простой интерфейс для разработки игр и прочих мультимедийных приложений.

**CI** (англ. Continuous Integration) - это практика разработки программного обеспечения, которая заключается в выполнении частых автоматизированных сборок проекта для скорейшего выявления и решения интеграционных проблем.

**Интерфейс** (англ. interface) - общая граница между двумя функциональными объектами, требования к которой определяются стандартом. Это совокупность средств, методов и правил взаимодействия между элементами системы.

**Игра GO -** логическая настольная игра с глубоким стратегическим содержанием, возникшая в Древнем Китае. Противники по очереди выставляют камни на перекрестья линий или пасуют. Цель игры — отгородить себе как можно больше свободных перекрестков. У каждого игрока достаточно камней, чтобы заполнить всю доску.

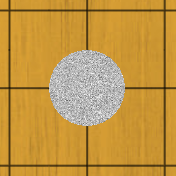
**Сетевая игра** (игра с сетевой структурой) - являеся разделом теории игр, который изучает как методы формирования связей между игроками в конфликтно-управляемых системах, так и правила определения выигрышей игроков с учётом этих связей. В данном случае формирование связей между игроками осуществляется за счет сети интернет

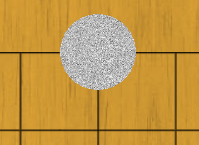
**Дыхания** (в рамках игры) — количество свободных перекрестков на доске, которые находятся рядом с камнем. Максимальное количество дыханий — 4.

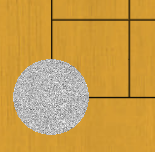
**ВВЕДЕНИЕ**

Рассмотрим основные правила этой игры. Игра Go происходит на специальных досках, которые обычно имеют размер 19 х 19, 13 х 13 или 9 х 9 линий. Игроки ходят по очереди, ставя камни белого и черного цветов на пересечения линий. Камень, или группа камней считается съеденной, если у него/неё не осталось дыханий. Съеденные камни идут в общий счёт. В конце игры выигрывает тот игрок, у которого оказывается наибольший счет. В счет идет количество съеденных камней и количество огороженных перекрестков. Также нельзя делать суицидальные ходы и ставить камень на то место, где он только что был съеден для того, чтобы съесть камень соперника.

Код данного проекта можно посмотреть по [ссылке](https://github.com/vasiliykadikov/go_game).

**Рисунок 1**. Расположение камня с четырьмя дыханиями

**Рисунок 2.** Расположение камня с тремя дыханиями

**Рисунок 3.** Расположение камня с двумя дыханиями

ТРЕБОВАНИЯ К ПРОЕКТУ

Данный проект должен удовлетворять следующим требованиям:

* отделение данных от логики (основной принцип ООП)
* наличие главного меню с возможностью выбора параметров игры
* возможность создавать лобби
* возможность присоединиться к лобби, зная его название
* автоматическое удаление камней
* реализация одновременной работы интерфейса и ожидания ответа с сервера за счёт многопоточности
* создание сервера, который автоматически будет отправлять информацию от одного пользователя к другому. А сами пользователи должны ее обрабатывать и отправлять новую информацию на сервер
* поддержка системы Continuous Integration (CI)
* наличие тестов google test

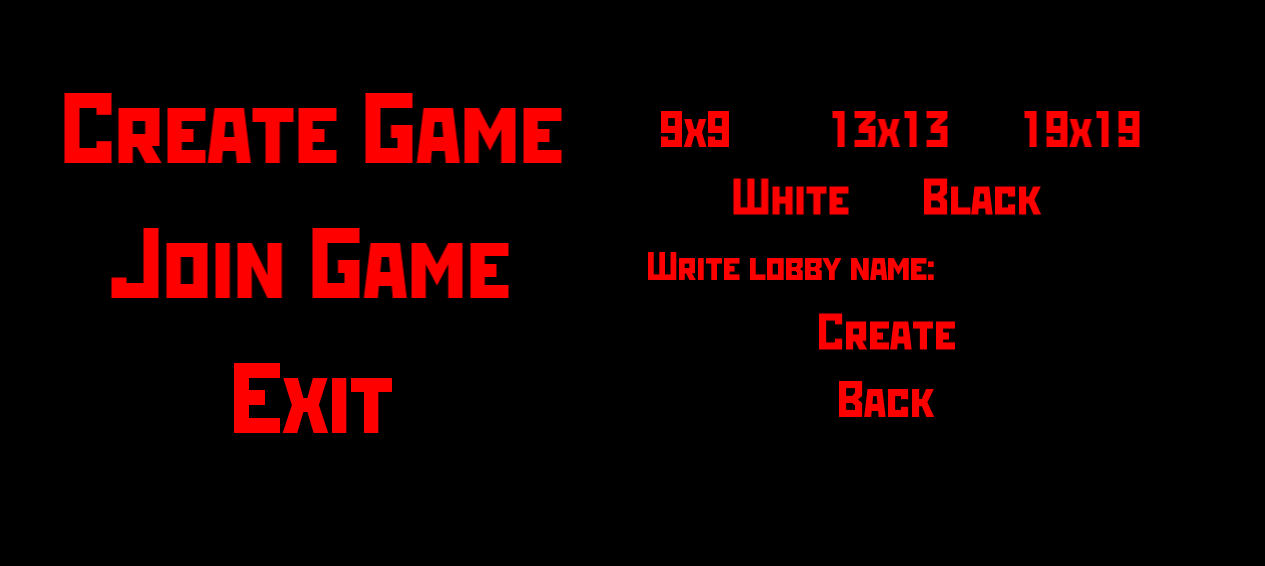
ПРОЕКТИРОВАНИЕ СИСТЕМЫ

Система данного приложения состоит из 4 классов и двух файлов с функциями:

* Класс “Кнопка” (Button)
* Класс “Игральный стол” (Table)
* Класс “Камень” (TableStone)
* Класс “Основное меню” (MainMenu)
* Функции для основного меню (for\_mainmenu)
* Математические функции (math\_functions)

**Button** – класс, необходимый для создания кнопок, которые будут иметь текстовое поле, поле и методы, необходимые для того, чтобы пользователь мог взаимодействовать с объектами этого класса посредством мыши и клавиатуры. Текст этого класса имеет шрифт *RodchenkoBTT.ttf,* находится в папке fonts.

**Table** – класс, объектом которого является игральный стол, хранящий данные о координатах пересечений линий, необходимых для определения мест для того, чтобы поставить камень, которые являются объектом класса TableStone.

**Рисунок 4.** Вид кнопок, создаваемых в классе Button

**TableStone –** класс, объектами которого являются игральные камни. Конструктор позволяет сознавать камни белого и черного цветов, радиус их зависит от размера игрального стола, а координаты зависят от координаты курсора, которые проверяются в классе MainMenu. Для того, чтобы камни не были чрезмерно однородные, в папке images находятся картинки, которые загружаются в поле текстур данного класса.

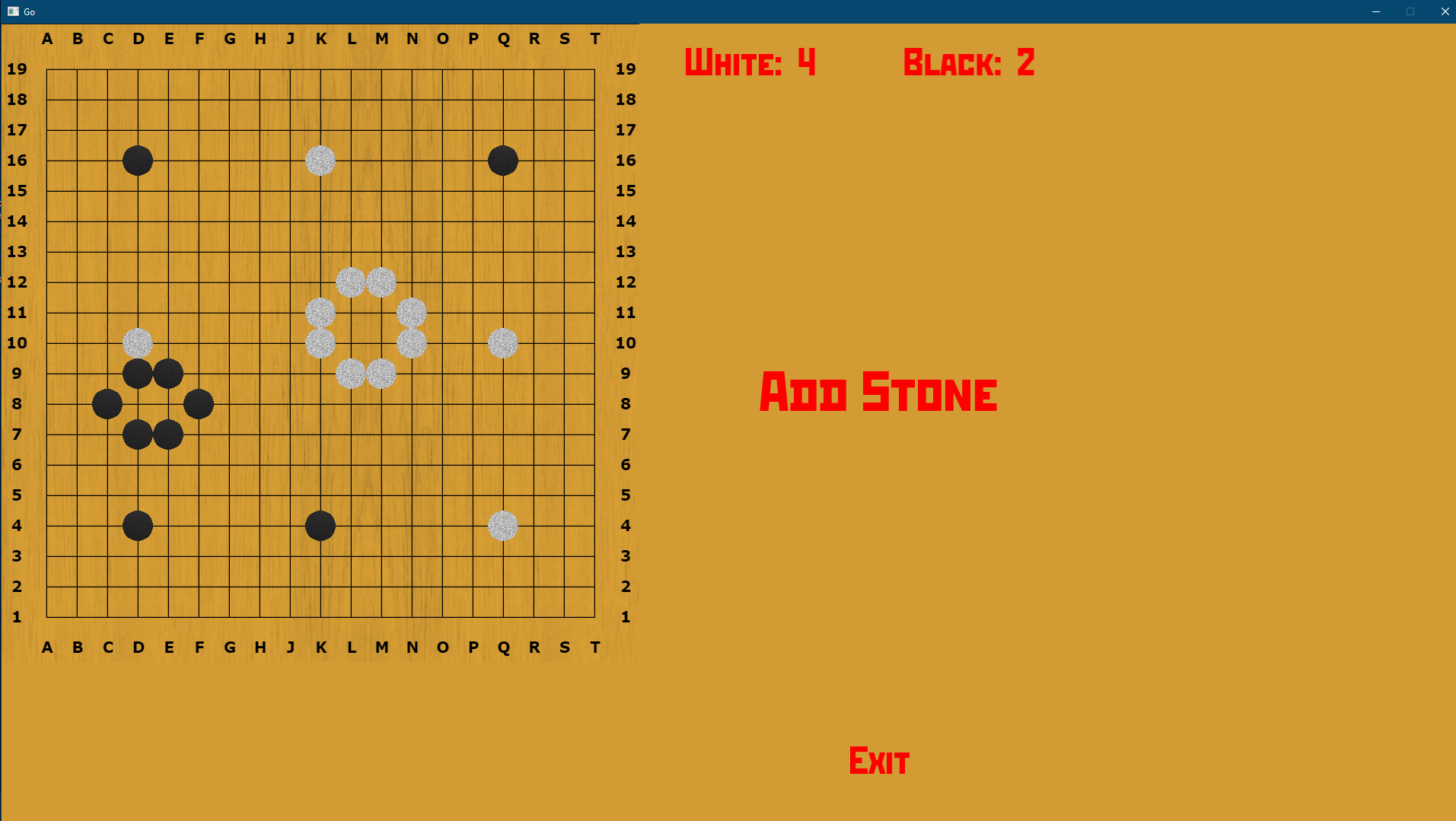
**MainMenu** – класс, объединяющий в себе логику, описанных выше классов. Имеет два основных цикла. Первый — с прорисовкой непосредственно главного меню. Пользователь имеет возможность либо создать лобби, либо присоединиться к нему, зная его название. Пользователь, создающий лобби выбирает размер доски и цвет своих камней.

Второй цикл включает в себя непосредственно процесс игры. Интерфейс включает в себя доску, камни, которые пользователи поставили на эту доску, счет съеденных камней, кнопку выхода из игры и кнопку добавить камень (то есть отправить сопернику добавленный камень).

Также этот класс включает в себя методы, необходимые для удаления камней, у которых не осталось дыханий. Программа для удобства создает так называемый фантомный камень на пересечении линий в соответствии с местоположением курсора, а при нажатии на кнопку мыши создает настоящий камень. Функции, помогающие определить где создавать камень в соответствии с местоположением курсора, находятся в класск Table, а также в файле math\_functions.

Сетевая часть реализована несколькими функциями, ожидающими подключения, а также отправляющими поставленные камни и ожидающие новых от соперника.

Кнопки, созданные в обоих циклах реагируют, меняя цвет при наведении на них курсора. Добиться этого помогает ряд функций, находящихся в самом классе MainMenu, а также функции, находящиеся в файле for\_mainmenu.

**Рисунок 5.** Вид игровой части программы

Сервер состоит из двух классов:

* Класс непосредственно сервера (server\_for\_go)
* Класс игрового лобби (playtable)

**server\_for\_go** – класс, реализующий сетевую часть сервера. Он отслеживает новые подключения, а также пакеты от уже подключенных пользователей и отправляет их сопернику, а также отслеживает какие пользователи отключились. Реализация сетевой части осуществляется за счет библиотеки SFML/Network. Эта библиотека содержит все необходимые для этого проекта методы. Также полем этого класса является вектор объектов класса playtable.

**playtable** – класс, основной задачей которого является хранение сокетов данной конкретной игры, чтобы можно было отправлять через сервер пакеты от одного пользователя к другому. Один из методов класса server\_for\_go в случае, если поступил пакет от какого-то пользователя, ищет сокет этого пользователя в векторе, хранящим сокеты игр, а после по второму сокету отправляет уже пакет второму игроку.

ВЫБОР ТЕХНОЛОГИЙ

**Выбор языка программирования**

Выбор языка программирования очень важная часть в разработке любого проекта. Нужно подобрать именно тот язык, который будет самым оптимальным для конкретной разработки. Именно поэтому чем больше проект, тем больше стек технологий, который в нем используется. Глобальные проекты состоят из огромного числа подпроектов. Для каждого из них выбирается тот язык, который будет наилучшем образом справляться с задачами, поставленными в нем.

Важными критериями при выборе технологий являются:

* Размер и тип проекта
* Сложность проекта
* Скорость разработки
* Доступные инструменты разработки
* Наличие готовых решений
* Гибкость решения
* Наличие подробной документации
* Требования к нагрузкам
* Требования к безопасности
* Кроссплатформенность
* Возможность интеграции с другими решениями

Так как объем данного проекта не очень большой, а время на разработку ограничено, то стоял вопрос выбора одного языка, а не нескольких.

Всем известно, что языки программирования разделяются по сфере применения. Основными сферами являются веб-разработка, мобильная и игровая разработка. Самыми популярными языками в веб-разработке на данный момент являются: HTML, CSS, JavaScript, Java , Python и PHP . В разработке мобильных приложений на Android: Java, на iOS: Swift и ObjectiveC. И наконец в разработке игр: С#, С++, JavaScript, Java, Smalltalk. Данный проект относится к сфере разработки игр, поэтому выбор был из последнего списка ЯП, технологии которого относятся к объектно-ориентированным языкам программирования.

Рассмотрев достоинства и недостатки каждого из перечисленных языков, выбор пал на С++. Ниже приведено подробное доказательство принятого решения.

С++ - язык общего назначения и задуман для того, чтобы настоящие программисты получили удовольствие от самого процесса программирования. За исключением второстепенных деталей он содержит язык С как подмножество. Язык С расширяется введением гибких и эффективных средств, предназначенных для построения новых типов. Программист структурирует свою задачу, определив новые типы, которые точно соответствуют понятиям предметной области задачи. Такой метод построения программы обычно называют абстракцией данных. Информация о типах содержится в некоторых объектах типов, определенных пользователем. С такими объектами можно работать надежно и просто даже в тех случаях, когда их тип нельзя установить на стадии трансляции. Программирование с использованием таких объектов обычно называют объектноориентированным. Если этот метод применяется правильно, то программы становятся короче и понятнее, а сопровождение их упрощается.

Достоинства С++:

* Чрезвычайно мощный язык, содержащий средства создания эффективных программ практически любого назначения.
* Компилируемость со статической типизацией.
* Сочетание высокоуровневых и низкоуровневых средств. 9
* Реализация ООП.
* Работает максимально быстро.
* Предсказуемое выполнение программ, что является важным для построения систем реального времени.
* Автоматический вызов деструкторов объектов при их уничтожении, причём в порядке, обратном вызову конструкторов. Это упрощает (достаточно объявить переменную) и делает более надёжным освобождение ресурсов (память, файлы, семафоры и т. п.), а также позволяет гарантированно выполнять переходы состояний программы, не обязательно связанные с освобождением ресурсов (например, запись в журнал).
* Пользовательские функции-операторы позволяют кратко и ёмко записывать выражения над пользовательскими типами в естественной алгебраической форме.
* Поддерживаются различные стили и технологии программирования, включая традиционное директивное программирование, ООП, обобщённое программирование, метапрограммирование (шаблоны, макросы).
* Используя шаблоны, возможно создавать обобщённые контейнеры и алгоритмы для разных типов данных, а также специализировать и вычислять на этапе компиляции.
* Используя шаблоны и множественное наследование можно имитировать классы-примеси и комбинаторную параметризацию библиотек. Такой подход применён в библиотеке Loki, класс SmartPrt которой позволяет, управляя всего несколькими параметрами времени компиляции, сгенерировать около 300 видов «умных указателей» для управления ресурсами.
* Кроссплатформенность: стандарт языка накладывает минимальные требования на ЭВМ для запуска скомпилированных программ. Для определения реальных свойств системы выполнения в стандартной библиотеке присутствуют соответствующие возможности (например, std::numeric\_limits ). Доступны компиляторы для большого количества платформ, на языке C++ разрабатывают программы для самых различных платформ и систем.
* Эффективность. Язык спроектирован так, чтобы дать программисту максимальный контроль над всеми аспектами структуры и порядка исполнения программы. Ни одна из языковых возможностей, приводящая к дополнительным накладным расходам, не является обязательной для использования — при необходимости язык позволяет обеспечить максимальную эффективность программы.
* Имеется возможность работы на низком уровне с памятью, адресами.
* Высокая совместимость с языком Си, позволяющая использовать весь существующий Си-код (код на Си может быть с минимальными переделками скомпилирован компилятором C++; библиотеки, написанные на Си, обычно могут быть вызваны из C++ непосредственно без каких-либо дополнительных затрат, в том числе и на уровне функций обратного вызова, позволяя библиотекам, написанным на Си, вызывать код, написанный на С++).

В совокупности с вышеописанным, можно сделать вывод, что в данной разработки программного обеспечения C++ - оптимальный выбор.

**Выбор используемых библиотек**

Существует много графических библиотек на С++. Основными являются OpenGL, QT, SFML и SDL.

В разработке данного программного обеспечения требовалась быстрая и простая библиотека. Так как у QT реализована очень медленная работа с графикой, она не рассматривалась как вариант для данного проекта.

По этой же причине не подходила библиотека SDL, которая помимо медленной работы не поддерживает концепцию ООП.

OpenGL в отличии от остальных чисто графическая библиотека. И так как в ней нет никаких средств для создания окна, ввода с клавиатуры и отрисовки кнопок, она не удовлетворяла требуемым условиям.

SFML – являлся самым оптимальным вариантом графической библиотеки в разработке данного проекта.

Преимущества SFML:

* Большой, очень простой и понятный фреймворк над разносторонними библиотеками.
* Имеет лицензию ‘zlib/png license’, что означает возможность использования в коммерческих целях.
* Наличие понятной документации с примерами.
* Обеспечивает простой интерфейс для различных компонентов ПК.
* Полностью открытый исходный код.
* Содержание ряда модулей ( Audio, Window, Graphics, Main, System) для простого программирования игр и мультимедиа приложений.
* Все библиотеки независимы друг от друга.
* Поддержка концепции объективно-ориентированного программирования
* Компиляция и запуск в самых распространенных операционных системах: Windows, Linux, Mac OS X.
* Имеет официальную привязку к языкам С.

ОПИСАНИЕ ТЕХНИЧЕСКИХ РЕШЕНИЙ

В ходе написания курсового проекта возникли следующие проблемы:

1. При введении названия лобби с клавиатуры при нажатии на клавишу одна буква пропечатывалась несколько раз из-за того, что время одного обхода цикла меньше, чем время нажатия пользователем на клавишу, из-за чего программа при каждом обходе цикла добавляет букву при нажатой клавише. Данная проблема решена путем добавления флага, который позволяет добавлять букву только при нажатии клавиши, а не при её удерживании.
2. Сложность в определении состояния disconnect для сокета. Решение данной проблемы таково, что при отправке пакета пользователю, можно узнать состояние сокета. В случае, если пользователь отключился, то программа отключает и второго пользователя.

ЗАКЛЮЧЕНИЕ

В процессе выполнения курсового проекта были получены навыки работы с графической библиотекой SFML, освоены новые и закреплены уже изученные принципы объектноориентированного программирования. Получен навык в написании алгоритмов (в частности алгоритма для удаления камней на доске). Также получены знания в области сетевого программирования. В результате была реализована сетевая игра Go.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Документация: Стандартные библиотеки С++ [Электронный курс]. URL: www.cplusplus.com/reference/ (дата обращения: 20.09.2020)
2. Документация: Официальная документация по языку программирования С++ [Электронный курс]. URL: http://ru.cppreference.com/w/ (дата обращения: 30.09.2020)
3. Документация: Официальная документация по графической библиотеке SFML [Электронный курс].

URL: https://www.sfml-dev.org/documentation/2.5.1/group\_\_graphics.php (дата обращения: 07.10.2020)

1. Уроки по SFML [электронный курс]. URL: <https://kychka-pc.ru/> (дата обращения: 07.10.2020)
2. Официальный репозиторий графической библиотеки SFML [Электронный курс]. URL: https://github.com/SFML/SFML/ (дата обращения: 20.10.2020)
3. Документация: Официальная документация по сетевой библиотеке SFML [Электронный курс].

URL: <https://www.sfml-dev.org/documentation/2.5.1/group__network.php>  
(дата обращения: 15.11.2020)