Учреждение образования
«Белорусский государственный технологический университет»

Факультет информационных технологий
Кафедра информационных систем и технологий
Специальность 1-40 05 01 “Информационные системы и технологии”

ОТЧЕТ
по учебной ознакомительной практике

Исполнитель

студент 1 курса 2 группы _____ Афанасенко Е.Д.
(подпись, дата) (Ф.И.О.)

Руководитель практики
от кафедры
Ассистент _____ Сазонова Д.В.
(должность, уч. звание) (подпись, дата) (Ф.И.О.)

Отчет защищен с оценкой_____

Минск 2025 г.

Оглавление

ВВЕДЕНИЕ

Данный проект представляет собой разработку настольной игры "Крестики-нолики" с использованием библиотеки SFML (Simple and Fast Multimedia Library). Основной целью является создание интуитивно понятного графического интерфейса для игры, поддерживающего несколько режимов: игру между двумя игроками на одном устройстве (PvP) и игру с искусственным интеллектом (PvBot).

Игра "Крестики-нолики" известна своей простотой, доступностью и популярностью среди пользователей всех возрастов. Благодаря этому проекту пользователи могут насладиться классической игрой в современном формате с улучшенной визуальной составляющей и удобным управлением.

Особенности данного проекта включают:

- Поддержку двух игровых режимов (игра с другом и игра с ботом).
- Гибкий графический интерфейс, адаптированный под разные состояния игры (выбор режима, выбор игрока, экран победы и т.д.).
- Анимации для улучшения визуального восприятия хода игры и результатов.
- Использование алгоритмов случайности и анализа для реализации ИИ, который действует как противник в режиме игры с ботом.
- Поддержка динамических эффектов, таких как подсветка кнопок при наведении мыши, анимации победной линии и др.

Проект ориентирован на использование базовых возможностей C++ и SFML, что делает его полезным примером для изучения данной библиотеки. Он подчеркивает аспекты разработки игр, включая работу с графикой, событиями, взаимодействием с пользователем и алгоритмами игрового процесса.

C++ — это универсальный объектно-ориентированный язык программирования высокого уровня. Он разработан для создания эффективного и производительного программного обеспечения.

SFML — это библиотека для работы с мультимедиа, графикой, аудио и сетями, созданная специально для разработки игр и приложений с графическим интерфейсом. Она предоставляет удобный интерфейс для работы с 2D-графикой, звуком, клавиатурой, мышью и другими устройствами ввода/вывода.

Visual Studio — это интегрированная среда разработки (IDE), разработанная корпорацией Microsoft. Она предназначена для написания, тестирования и отладки программ на множестве языков программирования, включая C++.

1. Постановка задачи

Цель разработки:

Создать приложение-игру "Крестики-нолики" с графическим интерфейсом, которое обеспечит интерактивность, удобство и визуальную привлекательность для пользователей.

Функциональные требования:

1. Режимы игры:
   - Игрок против игрока: Два пользователя играют друг против друга на одном устройстве.
   - Игрок против бота: Пользователь играет с программным соперником (ботом), который использует алгоритм для принятия решений.
2. Выбор настроек игры:
   - Возможность выбора символов для каждого игрока (например, X или O).
   - Интерактивное меню для выбора режима игры перед началом.
3. Анимация и графический интерфейс:
   - Плавная и привлекательная анимация для взаимодействий, таких как постановка символа, победа или ничья.
   - Чёткая визуализация игрового поля и состояния игры (например, подсветка победной линии).
4. Логика игры:
   - Проверка корректности ходов (например, невозможность поставить символ на занятое поле).
   - Автоматическое определение победы, поражения или ничьей.
   - Перезапуск игры без необходимости перезапуска приложения.
5. Дополнительные функции:
   - Отображение текущего хода (чей ход сейчас).

Нефункциональные требования:

1. Приложение должно быть разработано на языке программирования C++ с использованием библиотеки SFML.
2. Графический интерфейс должен быть удобным для пользователей и работать без задержек.
3. Приложение должно быть кроссплатформенным и запускаться на операционных системах Windows, macOS и Linux.

Ожидаемые результаты:

Созданное приложение позволит пользователям играть в "Крестики-нолики" с удобным и интуитивно понятным интерфейсом. Визуальные эффекты и анимации повысят вовлечённость и удовольствие от игры. Возможность игры против другого человека или искусственного интеллекта (бота), обеспечивая разнообразие сценариев и увеличивая аудиторию приложения. Пользователи смогут выбирать символы (крестик или нолик) и настраивать игровой режим, что создаёт ощущение индивидуального подхода. Приложение будет оптимизировано для работы на большинстве компьютеров благодаря использованию SFML, обеспечивая производительность даже на устройствах с минимальными ресурсами.

2. Обзор архитектуры

Архитектура игры "Крестики-нолики" построена на модульной основе, чтобы обеспечить простоту разработки, гибкость и расширяемость.

Класс TicTacToe

Ключевой элемент приложения, отвечающий за реализацию игровой логики и взаимодействие с графическим интерфейсом.

- **Ответственности:**
  - Управление состоянием игрового поля.
  - Обработка ходов игроков и определение исхода игры.
  - Взаимодействие с графическими элементами для отображения текущего состояния игры.
- **Основные структуры:**
  - std::array<Player, 9> — структура данных, представляющая игровое поле, состоящее из 9 клеток.
  - enum class Player — перечисление для обозначения текущего игрока (None, X, O).
  - enum class Mode — перечисление для управления игровым режимом (PvP, PvBot).

Графический интерфейс

Реализован с использованием библиотеки SFML для создания визуально привлекательного интерфейса.

- **Основные элементы:**
  - sf::RectangleShape — используется для создания кнопок (например, кнопки выбора режима) и линий (например, сетки игрового поля).
  - sf::Text — применяется для отображения текстовых элементов, таких как подсказки или сообщения о результате игры.
  - sf::Sprite и sf::Texture — обеспечивают загрузку и отображение изображений, таких как символы игроков или декоративные элементы.
- **Принципы работы:**
  - Элементы графического интерфейса организованы в сцены (меню, игровое поле).
  - Все графические объекты реагируют на пользовательский ввод (щелчки мыши, клавиши).

Режимы игры

- PvP (Player vs Player):

Позволяет двум игрокам соревноваться на одном устройстве. Управление игровыми ходами переключается между игроками.

- PvBot (Player vs Bot):

Пользователь играет против бота. Бот использует простой алгоритм, например, случайный выбор пустой клетки или более продвинутый — с базовой стратегией минимакса.

Анимация

Анимация добавляет динамичность и улучшает восприятие игры.

- **Элементы анимации:**

- Победные линии: После определения победителя соответствующая линия (горизонтальная, вертикальная или диагональная) плавно подсвечивается.
- Символы игроков: При постановке крестика или нолика используется эффект появления (например, увеличение символа из точки).

- **Реализация:**
  - Анимации реализуются с использованием временных интервалов (sf::Clock) для создания плавных переходов.
  - Эффекты настраиваются для достижения визуального комфорта и минимизации нагрузки на систему.

Взаимодействие между компонентами

1. Игровая логика (TicTacToe) обрабатывает пользовательский ввод и обновляет состояние игрового поля.
2. Графический интерфейс синхронизируется с логикой, отображая изменения (например, отрисовка символа или линии).
3. Режимы игры управляются на основе выбранного пользователем варианта и переключаются при необходимости.
4. Анимация активируется после определённых событий (ход, победа) для повышения визуального качества игры.

Эта архитектура позволяет эффективно обрабатывать ввод, обеспечивать стабильность логики и плавность графики.

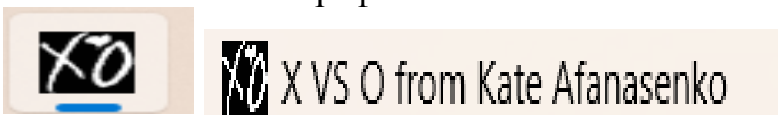

Рисунок 1- Лого игры

На рисунке 1 мы можем увидеть, самое первое окно игры, которое может увидеть пользователь запустив игру, ниже будет приведен код реализации данного окна.

```
if (!menuTexture.loadFromFile("C:/Users/user/MySFMLProject/x64/X
O.jfif")) {
     std::cerr << "Ошибка загрузки меню изображения\n";
 }
 menuSprite.setTexture(menuTexture);
```

Листинг 1 – Загрузка текстуры логотипа

Здесь текстура логотипа загружается из файла, а затем применяется к спрайту.

```
float scaleX = static_cast<float>(WINDOW_SIZE) /
menuTexture.getSize().x;
float scaleY = static_cast<float>(WINDOW_SIZE) /
menuTexture.getSize().y;
menuSprite.setScale(scaleX, scaleY);
```

Листинг 2 - Масштабирование логотипа под размер окна

Этот блок кода изменяет размер логотипа так, чтобы он соответствовал ширине окна.

```
if (gameMode == Mode::None) {
    window.draw(menuSprite);
```

Листинг 3 - Отображение логотипа в меню

Здесь логотип рисуется в окне игры, если пользователь находится в главном меню.

Рисунок 2 – Главное меню игры

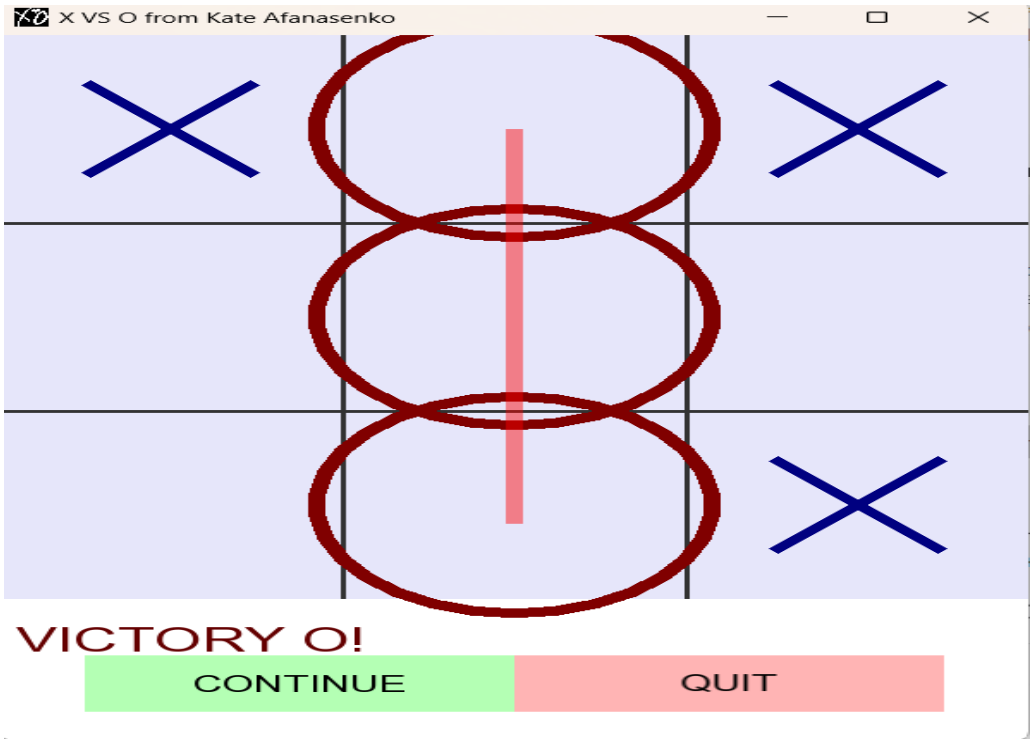

Рисунок 3 – Ход игры

На рисунках 2-3 мы можем увидеть основные экраны первой сцены, на которой игрок может выбрать с кем ему играть (с другом или с ботом), после выбора одной из этих кнопок нас перенесёт на поле на котором игрок сможет выбрать за кого он хочет играть: за крестик или за нолик, и после игры у нас появляется снизу надпись кто победил или же игра закончилась ничьёй и так же в виде кнопок опять предлагается выбор: закончить игру или же продолжить (если выбрать закончить-приложение закроется , если выбрать продолжить- игра обновится до момента выбора игроком за кого он хочет играть за x или за o). Так же для удобства каждая кнопка подсвечивается при наведение на неё курсора мыши.

Разберем код главного меню:

```
  // Кнопки выбора режима
  modePvPButton.setSize(sf::Vector2f(250, 60));
  modePvPButton.setFillColor(sf::Color(180, 255, 180));
  modePvPButton.setPosition(50, 20);

  modePvBotButton.setSize(sf::Vector2f(250, 60));
  modePvBotButton.setFillColor(sf::Color(180, 180, 255));
  modePvBotButton.setPosition(WINDOW_SIZE - 300, 20);

  modePvPText.setFont(font);
  modePvPText.setString("PLAY WITH A FRIEND");
  modePvPText.setCharacterSize(24);
  modePvPText.setFillColor(sf::Color::Black);
  centerTextInRect(modePvPText, modePvPButton.getGlobalBounds());

  modePvBotText.setFont(font);
  modePvBotText.setString("PLAY WITH BOT");
  modePvBotText.setCharacterSize(24);
  modePvBotText.setFillColor(sf::Color::Black);
  centerTextInRect(modePvBotText, modePvBotButton.getGlobalBounds());
// Рисуем меню выбора режима
if (gameMode == Mode::None) {
    window.draw(menuSprite);
    window.draw(modePvPButton);
    window.draw(modePvPText);
    window.draw(modePvBotButton);
    window.draw(modePvBotText);
    return;
}// Обработка кликов по кнопкам выбора режима
if (gameMode == Mode::None) {
    if
(modePvPButton.getGlobalBounds().contains(static_cast<sf::Vector2f>(mo
usePos))) {
        gameMode = Mode::PvP;
        choosingPlayer = true;
        gameStarted = false;
        reset();
        return;
```

Листинг 4 –Главное меню

3. Основные методы

Логика игры:

1. handleClick

Метод отвечает за обработку нажатий на игровом поле и кнопках управления. При нажатии определяет текущий ход, обновляет состояние игры, проверяет условия победы и переключает текущего игрока. Если включён режим игры против бота, автоматически выполняет ход за бота. Таким образом, **handleClick** является основным контроллером, связывающим игровую логику, интерфейс и взаимодействия пользователя.

2. reset

Сбрасывает состояние игры к начальным настройкам. Поле очищается, игроки возвращаются в исходное состояние, а индикатор конца игры обнуляется. Таким образом, метод **reset** гарантирует подготовку игры к новому началу, предоставляя игрокам чистый лист для дальнейших действий.

Рисование:

3. drawX и drawO

Методы для визуального отображения символов "X" и "O" на игровом поле. Используют графические функции для отрисовки символов на основе координат клеток. Эти методы обеспечивают интуитивное, четкое и эстетичное отображение символов, усиливая удовольствие от игры.

4. drawWinLine

Позволяет отобразить победную линию, соединяющую клетки, которые составляют выигрышную комбинацию. Этот метод также обеспечивает её анимацию или визуальное выделение.

5. drawAnimatedX и drawAnimatedO

Эти методы добавляют анимационные эффекты для символов "X" и "O". Они могут использоваться для особого отображения выигрышного символа или для улучшения общей визуализации.

Выбор режима:

6. gameMode и choosingPlayer

Отвечают за начальную настройку игры. **gameMode** определяет, будет ли игра в режиме для одного игрока (с ботом) или двух игроков. **choosingPlayer** задаёт, кто начинает игру — "X" или "O".

Бот:

7. Простая стратегия для хода бота

Бот выбирает случайную пустую клетку на игровом поле и выполняет ход. Алгоритм несложный, но достаточный для начального уровня игры. Если на поле остаётся единственная клетка, которая может привести к победе соперника, бот приоритетно занимает эту клетку, предотвращая проигрыш.

4. Результаты

1. Поддержка двух игровых режимов:
   - Режим игры против другого игрока: Два пользователя поочередно совершают ходы.
   - Режим игры против бота: Автоматизированный противник выполняет ходы, используя простую стратегию.
2. Анимации и визуальные эффекты:
   - Анимации символов "X" и "O" при появлении на игровом поле.
   - Отрисовка победной линии с анимацией, указывающей на выигрышную комбинацию.
   - Эффекты при переходе между ходами и завершении игры, такие как мигание или плавное изменение цвета клеток.
   - Удобный и стильный интерфейс, улучшающий пользовательский опыт.
3. Корректная логика проверки победы:
   - Механизм проверки победных комбинаций работает безошибочно.
   - Игра определяет выигрышные комбинации для строк, столбцов и диагоналей.
   - Реализована проверка ничьей, когда все клетки заполнены, но победная комбинация отсутствует.
4. Стабильность игры:
   - Реализована защита от ошибок ввода, исключающая повторные ходы в одну клетку.
   - При сбросе состояния все элементы интерфейса и логики корректно возвращаются в исходное положение.
5. Дружественность к пользователю:
   - Игра интуитивно понятна, с чёткими визуальными подсказками текущего игрока и состояния игры.
   - Простое и удобное управление для игроков любого уровня подготовки.
6. Перспективы развития:
   - Текущая реализация оставляет возможности для добавления новых функций, таких как более сложные стратегии для бота, дополнительные анимации или выбор тем оформления.

Эти результаты подчеркивают стабильность, удобство и эстетическую привлекательность проекта, делая игру интересной для широкой аудитории.

ЗАКЛЮЧЕНИЕ

Разработанная игра успешно демонстрирует возможности библиотеки **SFML** для создания 2D-игр с богатым графическим интерфейсом и анимациями. Игра реализует основные механики, такие как обработка событий, управление состоянием, анимация элементов, и обеспечивает интуитивно понятное взаимодействие пользователя с интерфейсом.

Код проекта структурирован и оптимизирован для удобства дальнейшего расширения. Это позволяет легко добавлять новые функции, такие как:

- Продвинутые стратегии для бота.
- Возможность выбора тем оформления.
- Улучшенные анимации и эффекты.
- Поддержка сетевой игры.

Игра является хорошим примером сочетания графики, логики и анимации, что делает её полезным проектом для обучения и демонстрации возможностей SFML. Она предоставляет игрокам комфортный и увлекательный опыт, оставляя широкие перспективы для дальнейшего развития.