МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение высшего образования

«Донецкий национальный технический университет»

ОТЧЁТ

по лабораторной работе №2

на тему: «Работа с ветками»

по курсу ««Профессиональная практика программной инженерии»

Выполнил: студент группы ПИ-20а

Рябцев М.К.

(подпись) «\_\_\_» \_\_\_\_\_\_\_2024г.

Приняла:

Незамова Л. В.

(подпись) «\_\_\_» \_\_\_\_\_\_\_2024г.

2024 г.

Ветка creative\_branch содержит файлы:

GPU.cpp – в котором находяться модуль GPU:

a. Модуль, эмулирующий работу центрального процессора японской консоли Sega Genesis.

b. Компонент, эмулирующий работу графического процессора Sega Genesis, отвечающий за вывод изображения на экран.

APU.cpp – в котором находяться модуль APU: Модуль, эмулирующий аудио процессор Sega Genesis, отвечающий за генерацию звука.

MMU.cpp – в котором находяться модуль Memory Management Unit (MMU): Компонент, отвечающий за управление доступом к памяти и эмуляцию работы памяти Sega Genesis.

Input Mapper.cpp – в котором находяться модуль Input Mapper: Модуль, отвечающий за эмуляцию взаимодействия с игровым контроллером Sega Genesis.

ROM.cpp – в котором находяться модуль ROM Loader: Компонент, обеспечивающий загрузку ROM-файлов игр и их эмуляцию.

Sound Blaster.cpp – в котором находяться модуль Sound Blaster Emulator: Модуль, эмулирующий звуковую карту Sound Blaster для улучшения качества звука.

Save.cpp – в котором находяться модуль Save State Manager: Компонент, отвечающий за сохранение и восстановление текущего состояния игры.

Debugging Tools.cpp – в котором находяться модуль Debugging Tools: Набор инструментов для отладки и тестирования эмулятора.

Ветка Lib2 содержит файлы:

Video Renderer.cpp – в котором находяться модуль Video Renderer: Модуль, отвечающий за отображение графики Sega Genesis на экране компьютера.

Input Recorder.cpp – в котором находяться модуль Input Recorder: Компонент, позволяющий записывать и воспроизводить ввод с контроллера для повторного тестирования игр.

Cheat Code.cpp – в котором находяться модуль Cheat Code Manager: Модуль для управления чит-кодами и их эмуляции.

BIOS.cpp – в котором находяться модуль BIOS Emulator: Компонент, эмулирующий работу BIOS Sega Genesis.

Game Genie Emulator.cpp – в котором находяться модуль Game Genie Emulator: Модуль для эмуляции расширений Game Genie и обеспечения поддержки чит-кодов.

Network Play.cpp – в котором находяться модуль Network Play: Компонент, позволяющий играть по сети с другими пользователями.

Frame Limiter.cpp – в котором находяться модуль Frame Limiter: Модуль, ограничивающий частоту обновления кадров для эмуляции реального времени.

Language Localization.cpp – в котором находяться модуль Language Localization: Компонент, позволяющий переводить интерфейс эмулятора на разные языки.

Ветка Laba2 содержит файлы:

Configuration Manager.cpp – в котором находяться модуль Configuration Manager: Модуль, позволяющий настраивать параметры эмулятора и сохранять их для последующего использования.

GUI.cpp – в котором находяться модуль GUI Controller: Компонент, отвечающий за создание графического интерфейса пользователя для удобного управления эмулятором.

Auto-Update Manager.cpp – в котором находяться модуль Auto-Update Manager: Модуль для автоматического обновления эмулятора до последней версии.

Save File Converter.cpp – в котором находяться модуль Save File Converter: Инструмент, предназначенный для изменения формата сохраненных файлов игр.

Netplay Lobby.cpp – в котором находяться модуль Netplay Lobby: Модуль для управления комнатами и создания сетевых игровых сессий.

Rewind Feature.cpp – в котором находяться модуль Rewind Feature: Функция, позволяющая перематывать время в игре для повторного прохождения уровней или исправления ошибок.

Game Database.cpp – в котором находяться модуль Game Database: База данных, содержащая информацию о всех играх Sega Genesis, совместимых с эмулятором.

Screen Capture.cpp – в котором находяться модуль Screen Capture: Инструмент для создания снимков экрана во время игры и сохранения их в различных форматах.

Программный код для каждого из модулей:

1. GPU.cpp (рис. 1)

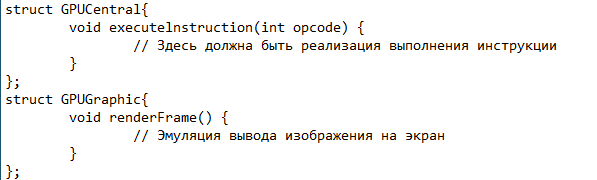


Рисунок 1 – GPU.cpp

1. APU.cpp (рис. 2)

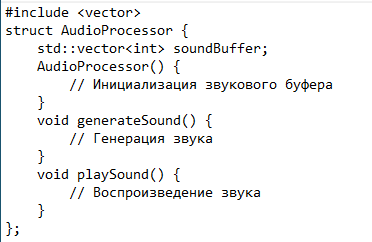


Рисунок 2 – APU.cpp

1. MMU.cpp (рис. 3)

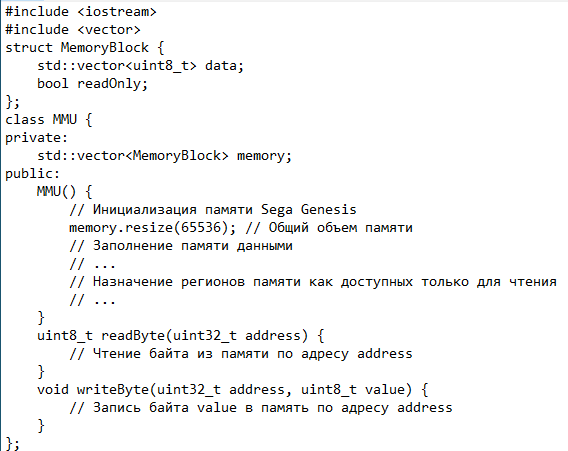


Рисунок 3 – MMU.cpp

1. Input Mapper.cpp (рис. 4)

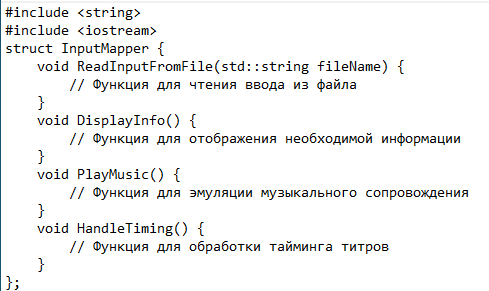


Рисунок 4 – Input Mapper.cpp

1. ROM.cpp (рис. 5)

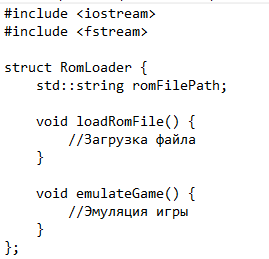


Рисунок 5 – ROM.cpp

1. Sound Blaster.cpp (рис. 6)

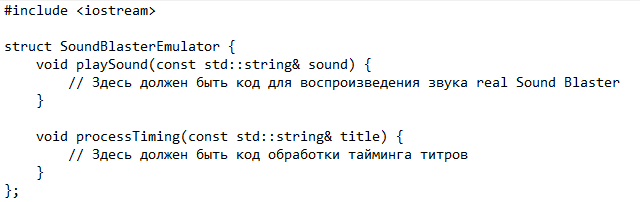


Рисунок 6 – Sound Blaster.cpp

1. Save.cpp (рис. 7)

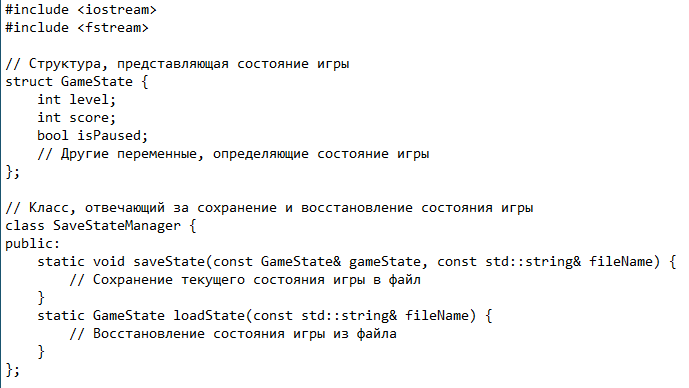


Рисунок 7 – Save.cpp

1. Debugging Tools.cpp (рис. 8)

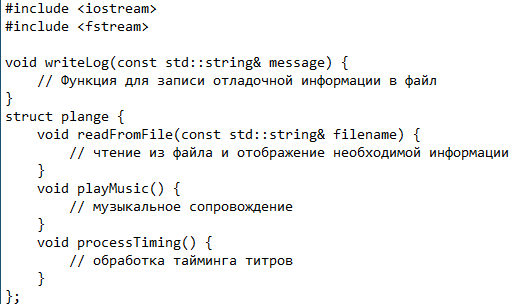


Рисунок 8 – Debugging Tools.cpp

1. Video Renderer.cpp (рис. 9)

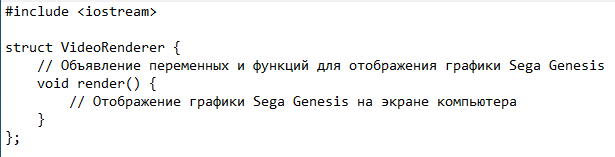


Рисунок 9 – Video Renderer.cpp

1. Input Recorder.cpp (рис. 10)

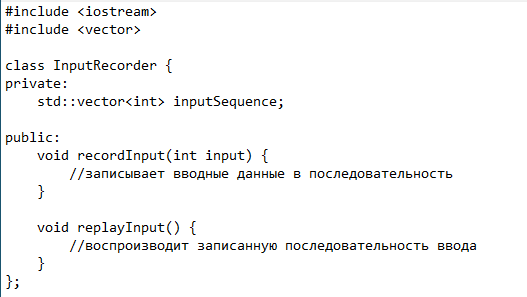


Рисунок 10 – Input Recorder.cpp

1. Cheat Code.cpp (рис. 11)

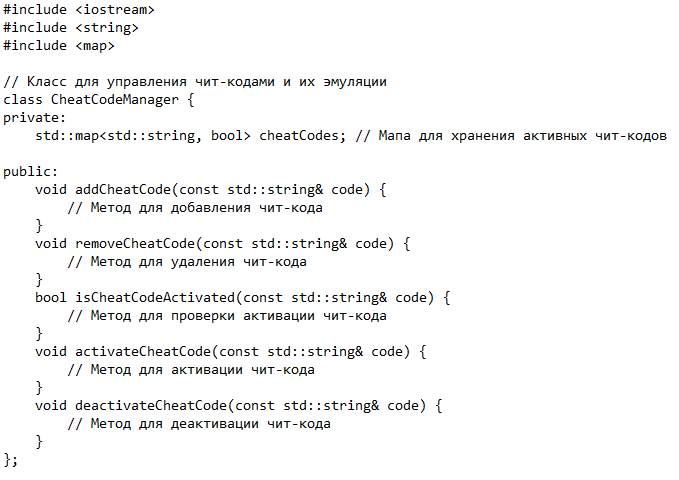


Рисунок 11 – Cheat Code.cpp

1. BIOS.cpp (рис. 12)

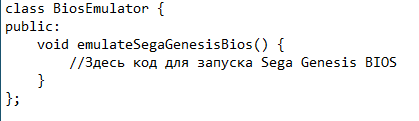


Рисунок 12 – BIOS.cpp

1. Game Genie Emulator.cpp (рис. 13)

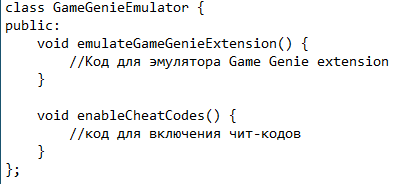


Рисунок 13 – Game Genie Emulator.cpp

1. Network Play.cpp (рис. 14)

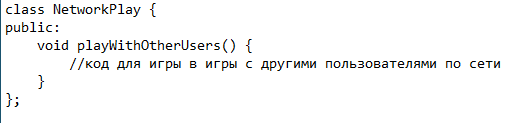
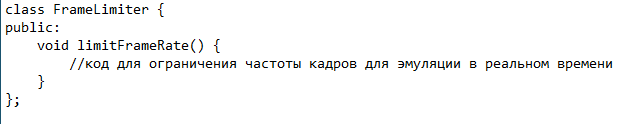


Рисунок 14 – Network Play.cpp

1. Frame Limiter.cpp (рис. 15)

Рисунок 15 – Frame Limiter.cpp

1. Language Localization.cpp (рис. 16)

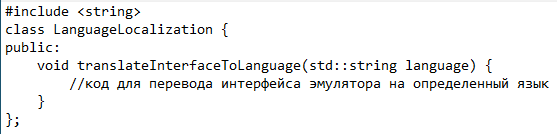


Рисунок 16 – Language Localization.cpp

1. Configuration Manager.cpp (рис. 17)

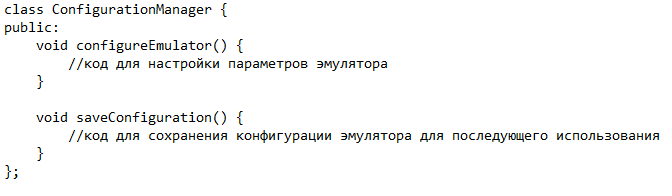


Рисунок 17 – Configuration Manager.cpp

1. GUI.cpp (рис. 18)

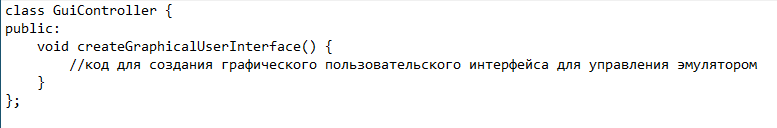


Рисунок 18 – GUI.cpp

1. Auto-Update Manager.cpp (рис. 19)

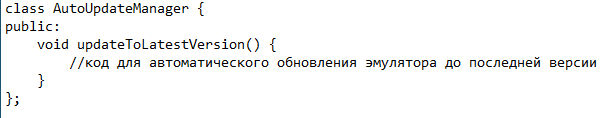
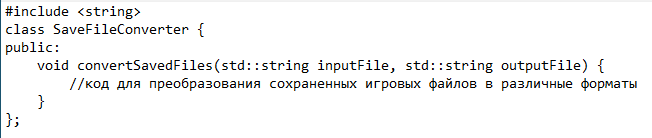


Рисунок 19 – Auto-Update Manager.cpp

1. Save File Converter.cpp (рис. 20)

Рисунок 20 – Save File Converter.cpp

1. Netplay Lobby.cpp (рис. 21)

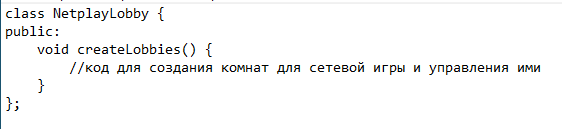


Рисунок 21 – Netplay Lobby.cpp

1. Rewind Feature.cpp (рис. 22)

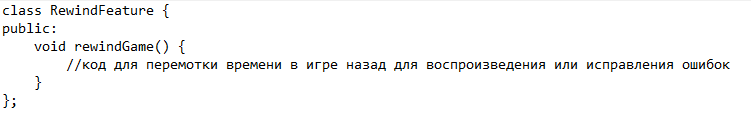


Рисунок 22 – Rewind Feature.cpp

1. Game Database.cpp (рис. 23)

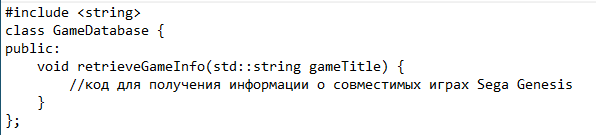


Рисунок 23 – Game Database.cpp

1. Screen Capture.cpp (рис. 24)

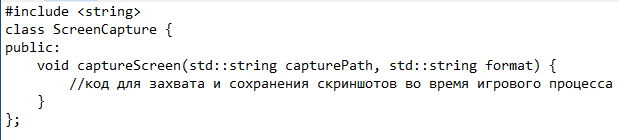


Рисунок 24 – Screen Capture.cpp

Создаем ветку creative\_branch и переходим в неё (рис. 25)

Команда “git checkout -b <навзание ветки>” создает новую ветку и переключается на нее. То есть, эта команда одновременно создает новую ветку и переключает Вас на нее, чтобы Вы могли начать работу в этой ветке.

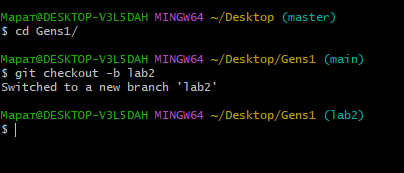


Рисунок 25 – Создание ветки lab2

Создаем в ветке файлы и ставим коммит, передаем файлы (рис. 26)

Команда `git add` используется для добавления изменений файлов в индекс (staging area) перед коммитом. Команда `git status` показывает текущее состояние репозитория, включая измененные файлы, файлы в индексе и другую информацию о состоянии репозитория.

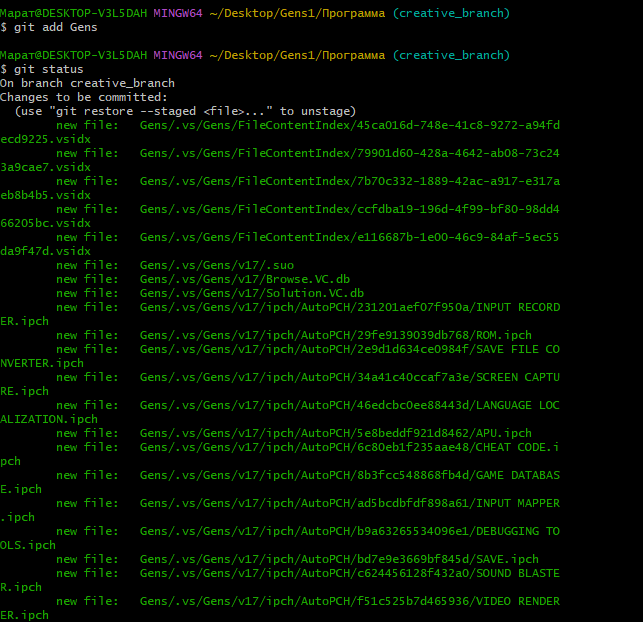
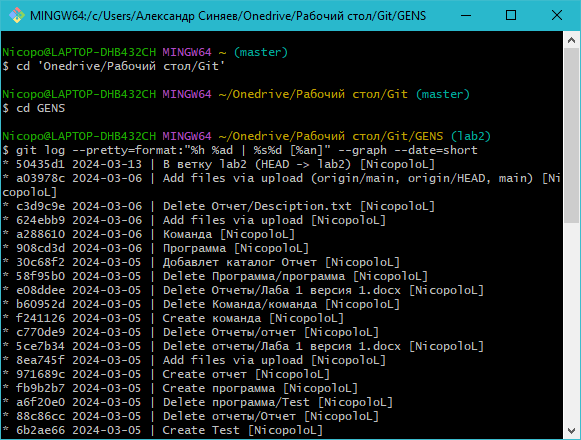


Рисунок 25 – Создаем в ветке файлы и ставим коммит

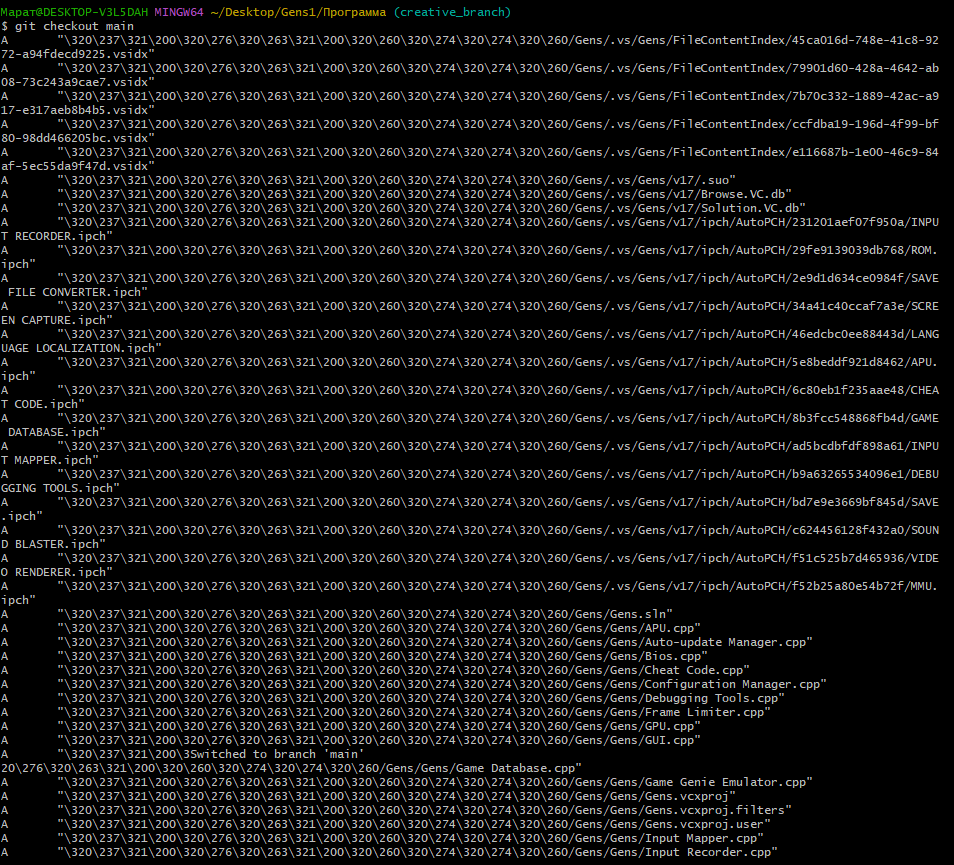
Вывод лог коммитов (рис. 26)

Команда git log --pretty=format:\"%h %ad | %s %d [%an]\" --graph --date=short выводит историю коммитов в Git с определенным форматированием.

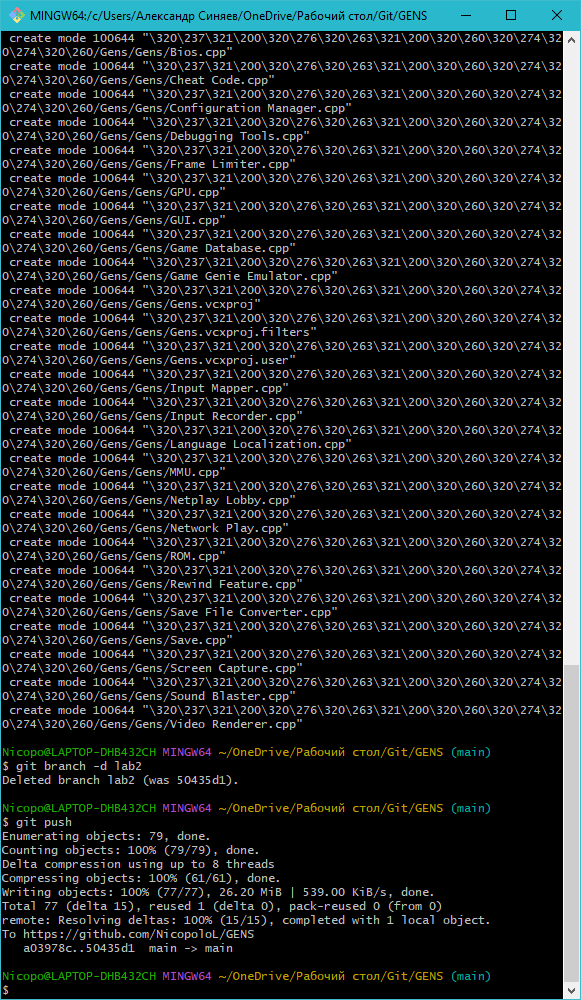
git Рисунок 26 – Вывод лог коммитов

Переключаем рабочую копию на main и передаем изменения с ветки, а далее удаляем ветку

Команда git checkout в Git используется для переключения между ветками или восстановления файлов из репозитория. Команда git merge используется для слияния изменений из одной ветки в другую. Она объединяет изменения из указанной ветки в текущую ветку. Команда git branch возвращает список всех веток проекта (локальных и удаленных) и помечает текущую активную ветку.







Ссылка <https://github.com/MaratRyabcev/Gens1.git>