МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение высшего образования

«Донецкий национальный технический университет»

Факультет ИСП

Кафедра ПИ им Л.П.Фельдмана

Лабораторная работа № 2

на тему: «Работа с ветками»

по дисциплине «Профессиональная практика программной инженерии»

Выполнил: студент группы ПИ-20а

\_\_\_\_\_\_\_\_\_\_\_\_\_\_Синяева Э.В.

(подпись) «\_\_\_» \_\_\_\_\_\_\_2021г.

Принял:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Незамова Л.В.

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

2024г.

**Цель работы:** познакомиться с основами использования веток в системе контроля версий Git.

Вариант 17. Эмулятор портативных устройств (GameBoy, PSP, Pokemon Mini).

1 Описать имеющиеся ветки и какие файлы реализовывают какой конкретно модуль и в какой ветке находятся.

**Модуль 1 (module1)** Ветка, содержащая модули (Рисунок 1):

Файлы (Содержимое файлов Приложение 1):

* Эмуляция аппаратной части - hardware\_emulation.py
* Графический интерфейс пользователя - user\_interface.py -
* Управление настройками - settings\_management.py
* Совместимость с играми - game\_compatibility.py
* Сохранение прогресса - progress\_saving.py
* Интеграция с онлайн-сервисами - online\_integration.py
* Улучшение графиков и звука - graphics\_sound\_enhancement.py

Фиксация изменений: "Добавлены изменения для модуля 1"

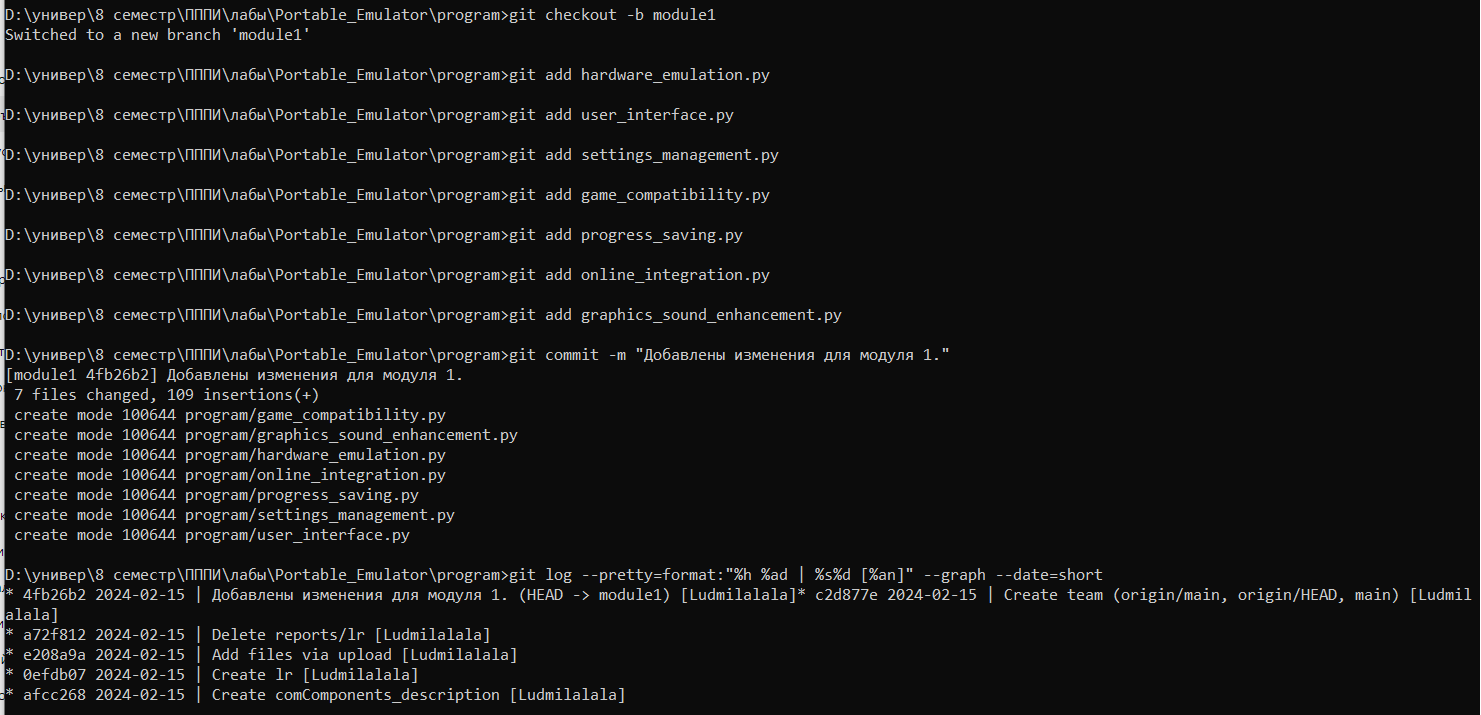


Рисунок 1 - Ветка module1

Вывод команды git log для ветки module1 (Рисунок 2)

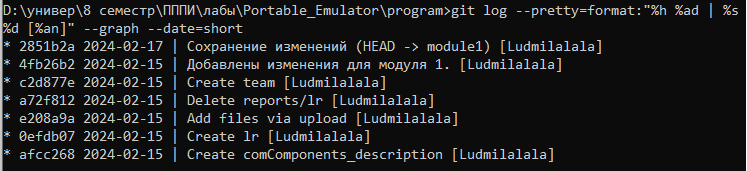


Рисунок 2 - Вывод команды git log

**Модуль 2 (module2)** Ветка, содержащая модули (Рисунок 3)

Файлы (Содержимое файлов Приложение 1):

* Поддержка дополнительных функций - additional\_features.py
* Обновления и поддержка - updates\_support.py
* Безопасность и конфиденциальность - security\_privacy.py
* Интеграция с облачными сервисами - cloud\_integration.py
* Геймпад-эмуляция - gamepad\_emulation.py
* Модуль управления эмуляцией - emulation\_control.py
* Поддержка мультиязычности - multilanguage\_support.py

Фиксация изменений: " Добавлены изменения для модуля 2"

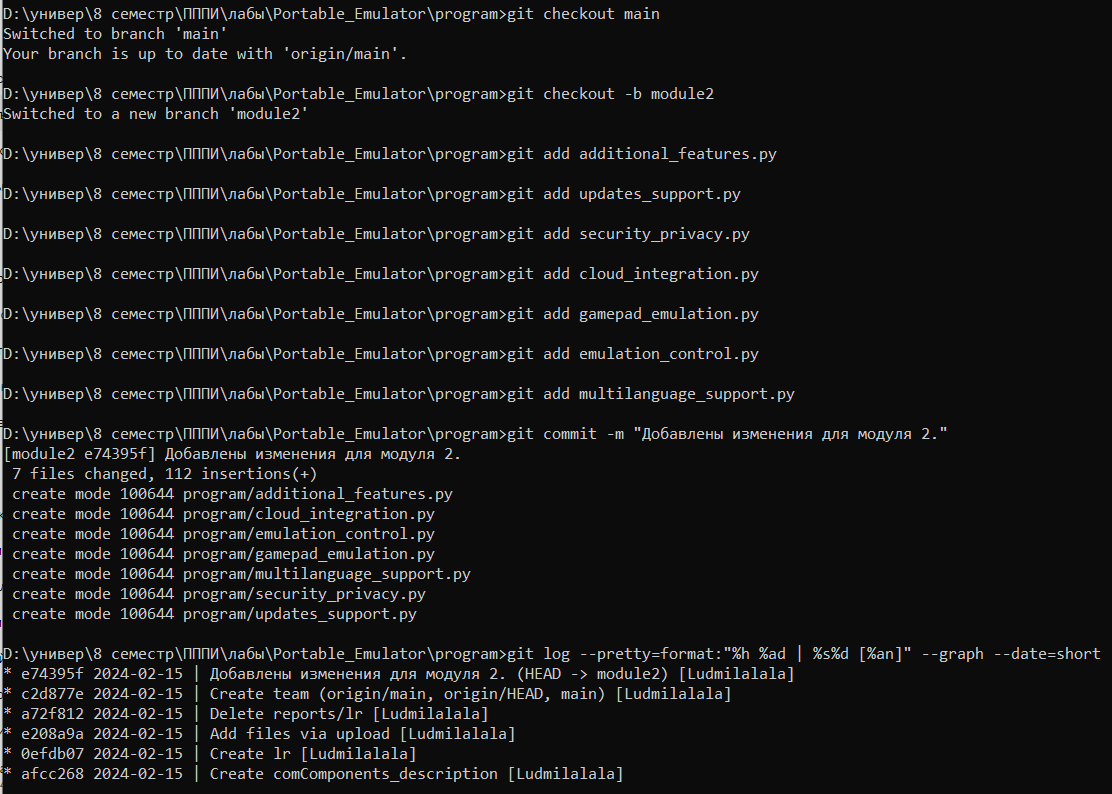


Рисунок 3 - Ветка module2

Вывод команды git log для ветки module2 (Рисунок 4)

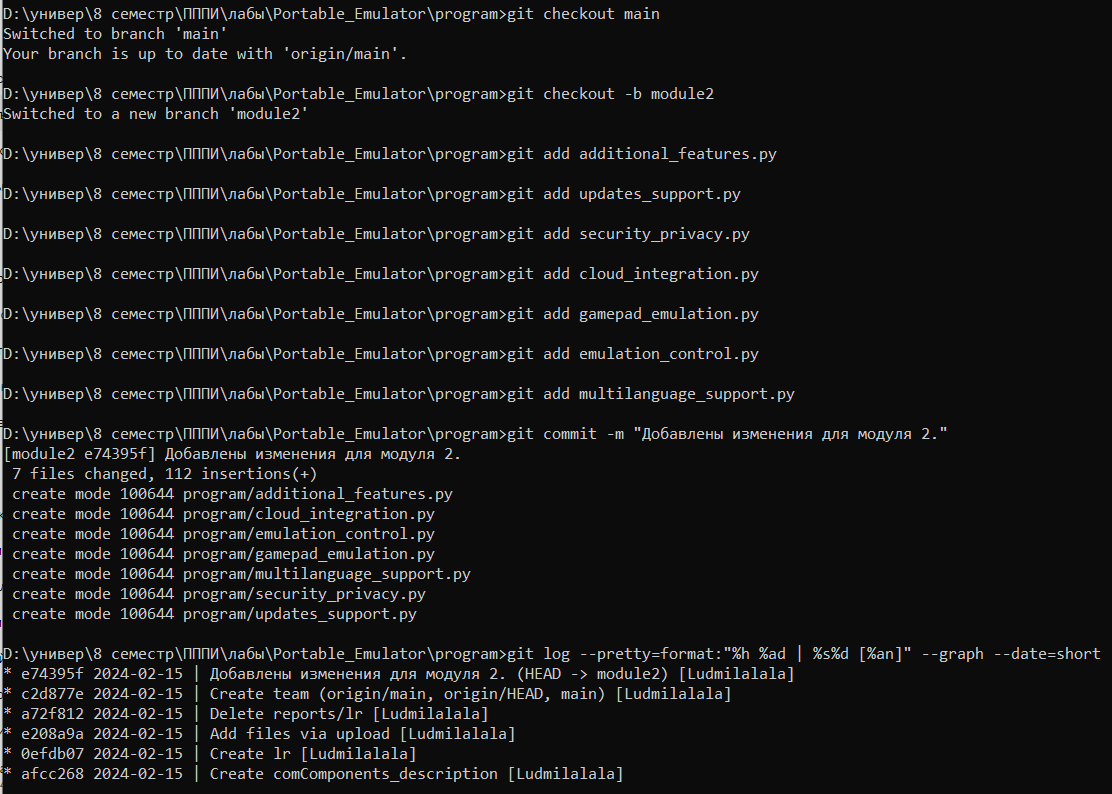


Рисунок 4 - Вывод команды git log

**Модуль 3 (module3)** Ветка, содержащая модули (Рисунок 5)

Файлы (Содержимое файлов Приложение 1):

* Автоматическое обновление игр - game\_auto\_update.py
* Поддержка сохранения и загрузки процесса игры - game\_progress\_management.py
* Поддержка различных форматов игровых образов - game\_formats\_support.py
* Возможность записи игрового процесса - gameplay\_recording.py
* Поддержка мобильных устройств - mobile\_device\_support.py
* Интеграция с настройками сетей - network\_settings\_integration.py
* Расширенные настройки графики и звука - advanced\_graphics\_audio\_settings.py
* Режимы эмуляции специальных устройств - special\_devices\_emulation.py

Фиксация изменений: " Добавлены изменения для модуля 3"

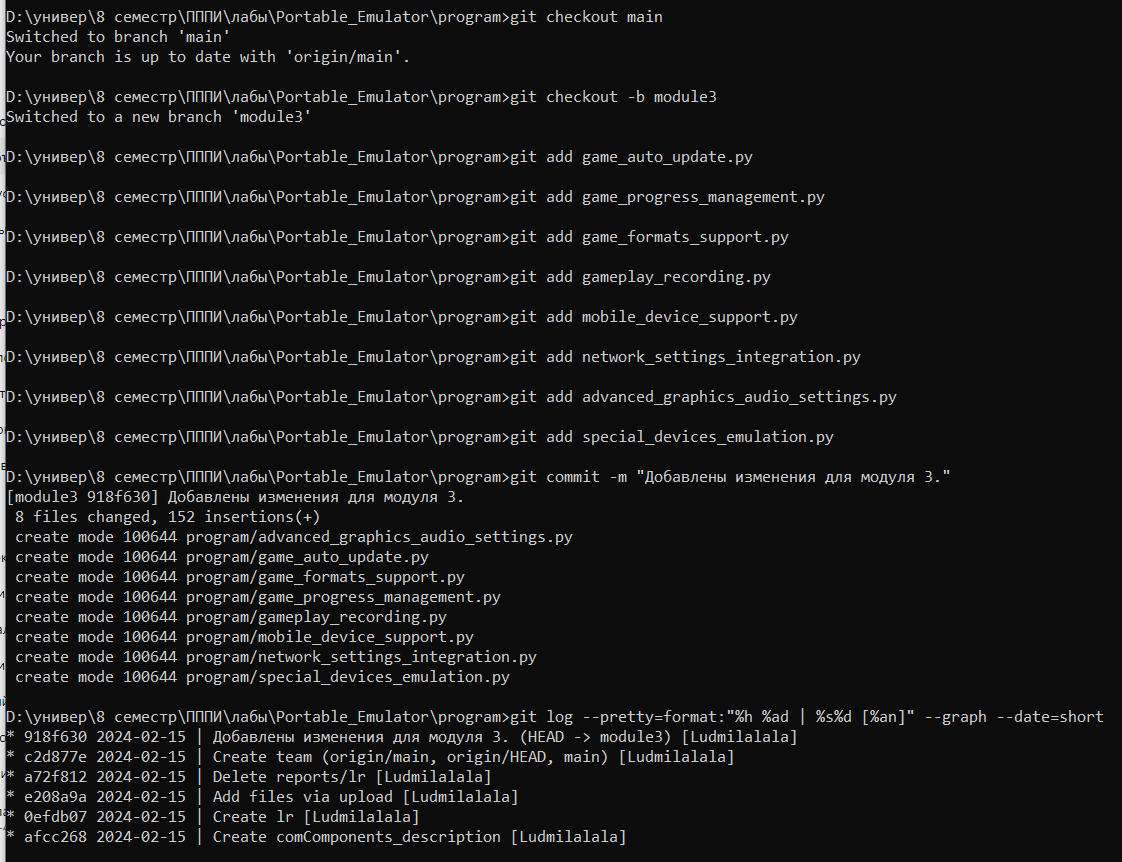


Рисунок 5 - Ветка module3

Вывод команды git log для ветки module3 (Рисунок 6)

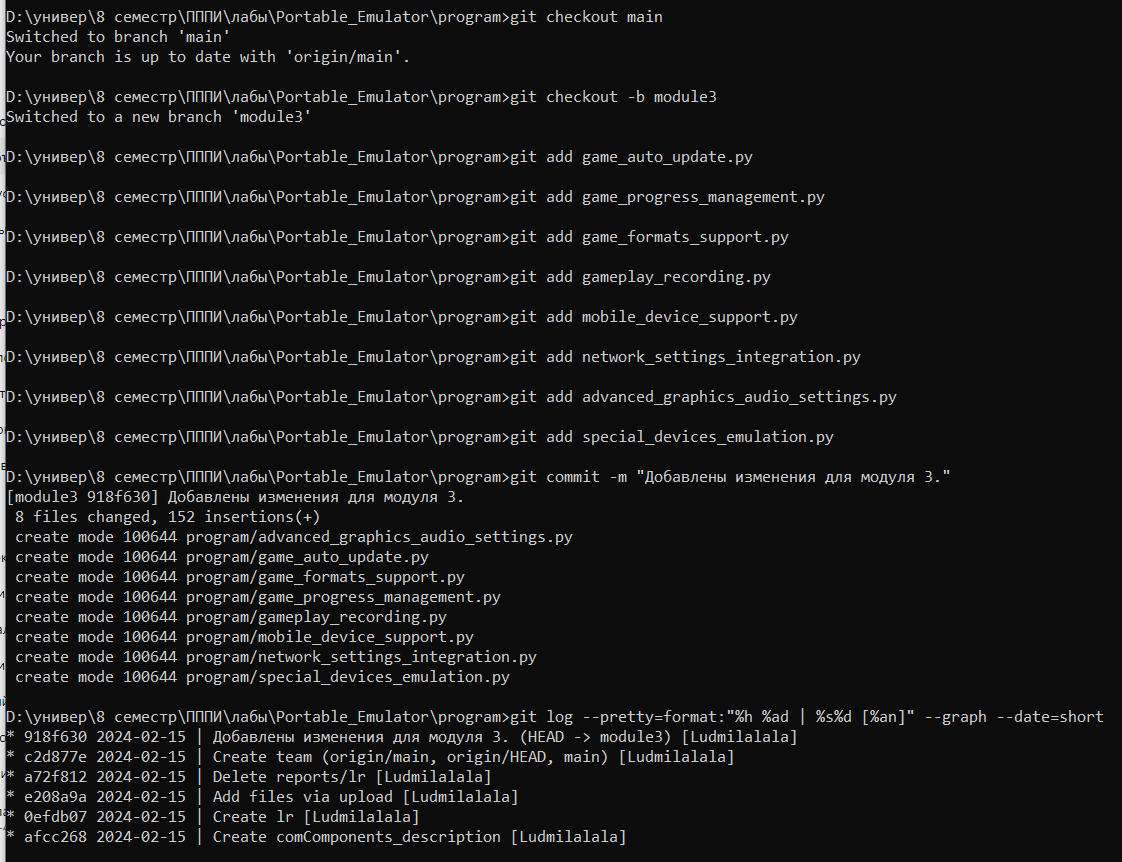


Рисунок 6 - Вывод команды git log

*Использованные команды:*

**git checkout -b**: Создание новой ветки и переключение на неё. Эта команда одновременно создает и переключается на новую ветку.

**git add**: Добавление файла в индекс для подготовки к коммиту.

**git commit -m** " ": Фиксация изменений в файлах, добавленных их индекса с предоставленным сообщением коммита.

**git commit -am** " ": Фиксация всех изменений в файлах, которые уже отслеживаются Git, с предоставленным сообщением коммита. Параметр -a автоматически добавляет все отслеживаемые файлы к коммиту.

**git checkout**: Переключение на существующую ветку.

**--pretty=format:"%h %ad | %s%d [%an]"**: Это опция форматирования вывода, которая определяет, какие данные о коммитах будут отображаться. Здесь **%h** отображает сокращенный хэш коммита, **%ad** - дату коммита, **%s** - сообщение коммита, **%d** - информацию о ветках и тегах, **%an** - имя автора коммита.

**--graph**: Это опция, которая добавляет графическое представление истории коммитов с использованием ASCII-графики.

**--date=short**: Это опция, которая устанавливает формат отображения даты в коротком формате (YYYY-MM-DD).

Файлы, присутствующие в репозитории (Рисунок 7 – Рисунок 8).

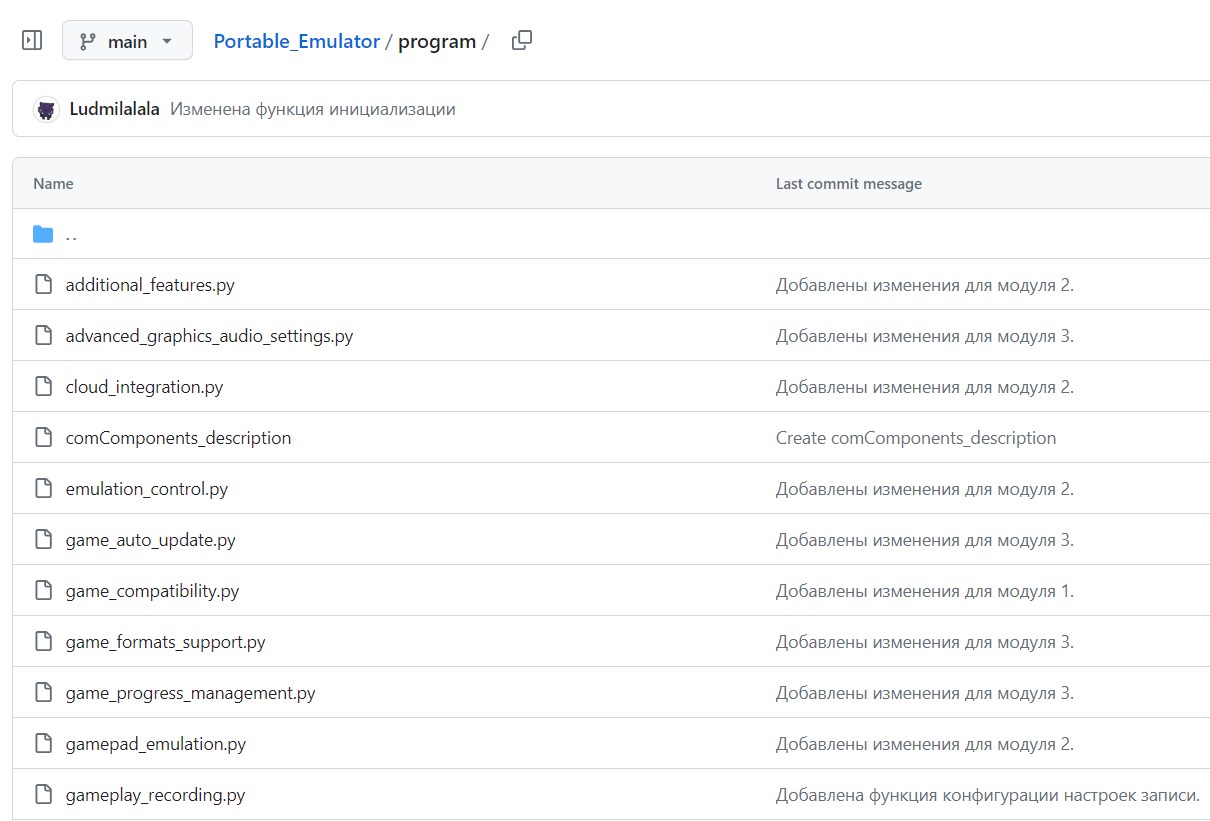


Рисунок 7 – Файлы в репозитории

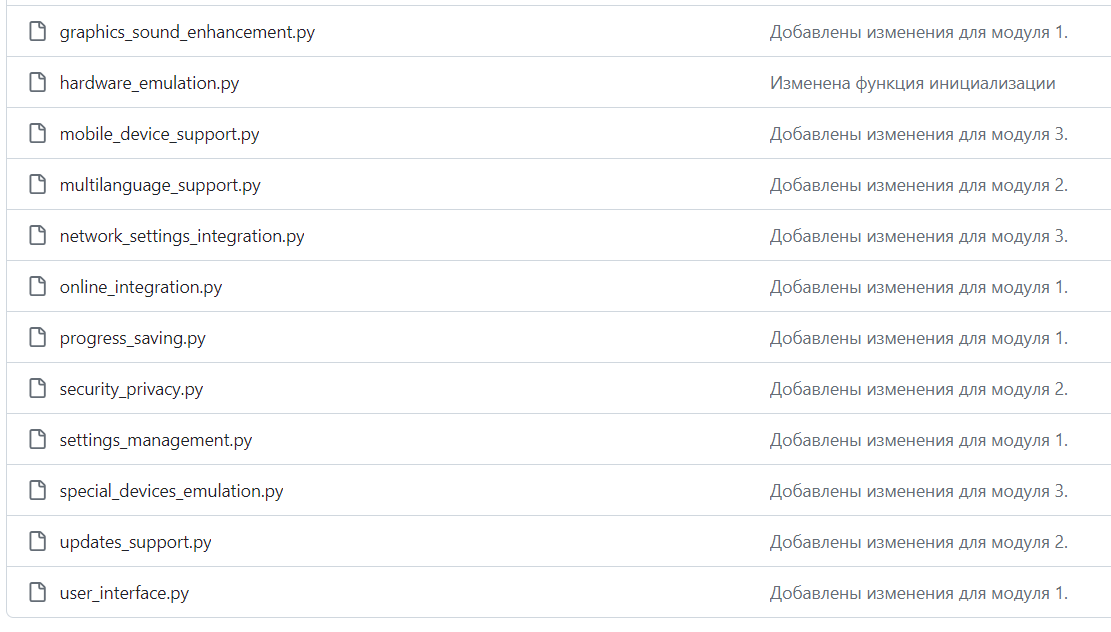


Рисунок 8 – Файлы в репозитории

2 Влить все ветки в master.

В некоторых репозиториях основная ветка может называться **main**, а не **master**.

Чтобы влить все созданные ветки в ветку **main**, необходимо выполнить следующие команды:

Переключиться на ветку **main**

**git checkout main -** эта команда переключается на ветку **main** в локальном репозитории.

Влить каждую ветку в **main** с помощью команды **git merge**

**git merge** – эта команда объединяет изменения из одной ветки в текущую ветку (в данном случае, ветку **main**).

Слияние ветки module1 в основную ветку (Рисунок 9).

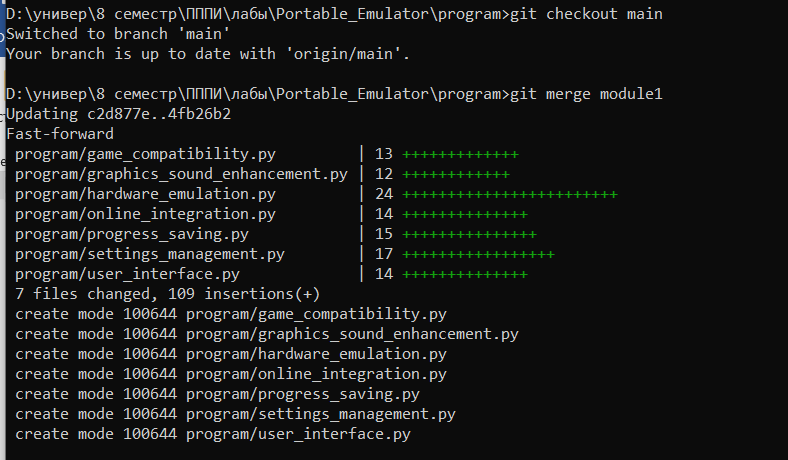


Рисунок 9 - Слияние ветки module1

Слияние ветки module2 в основную ветку (Рисунок 10).

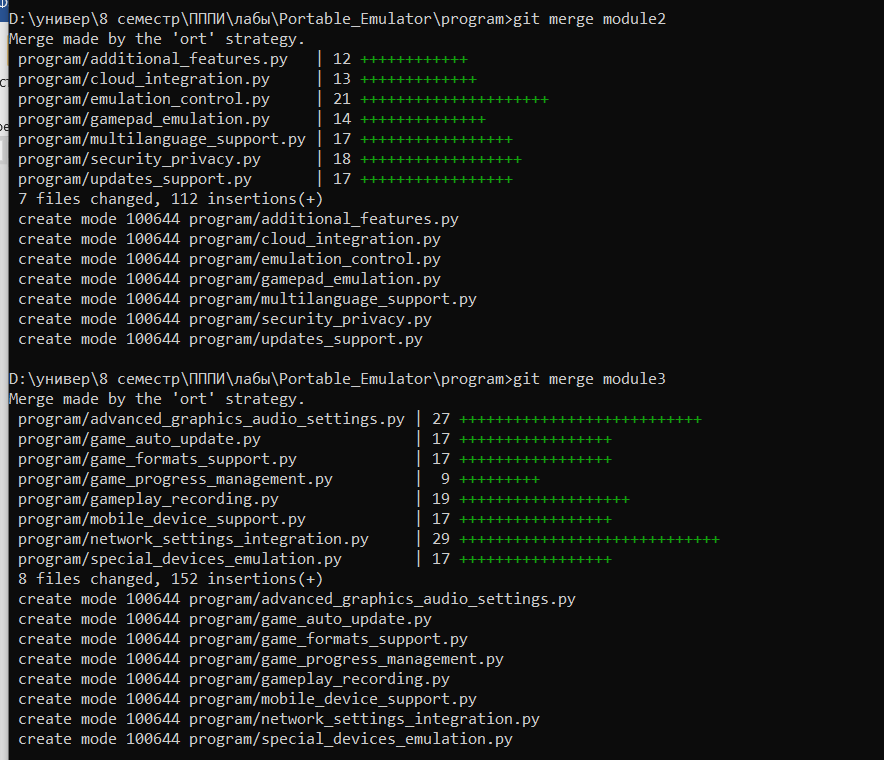


Рисунок 10 - Слияние ветки module2

Слияние ветки module3 в основную ветку (Рисунок 11).

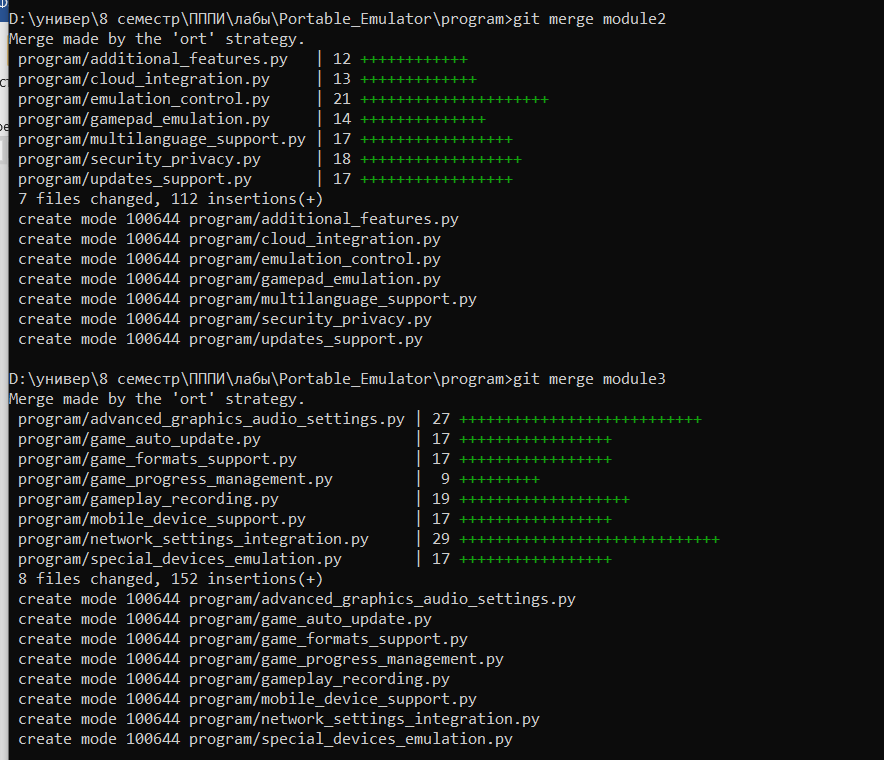


Рисунок 11 - Слияние ветки module3

Используем функцию **git commit -m** для фиксации изменений в файлах, добавленных их индекса с предоставленным сообщением коммита (Рисунок 12).

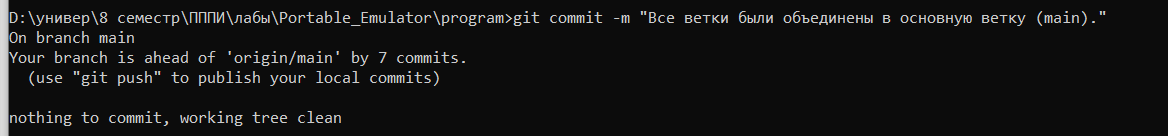


Рисунок 12 - Фиксация изменений

Вывод команды git log для trunk после вливания всех веток (Рисунок 13).

**trunk**: Это название ветки, для того чтобы просмотреть историю коммитов.

Нужно заменить trunk на фактическое название вашей основной ветки, в моем случае **main.**

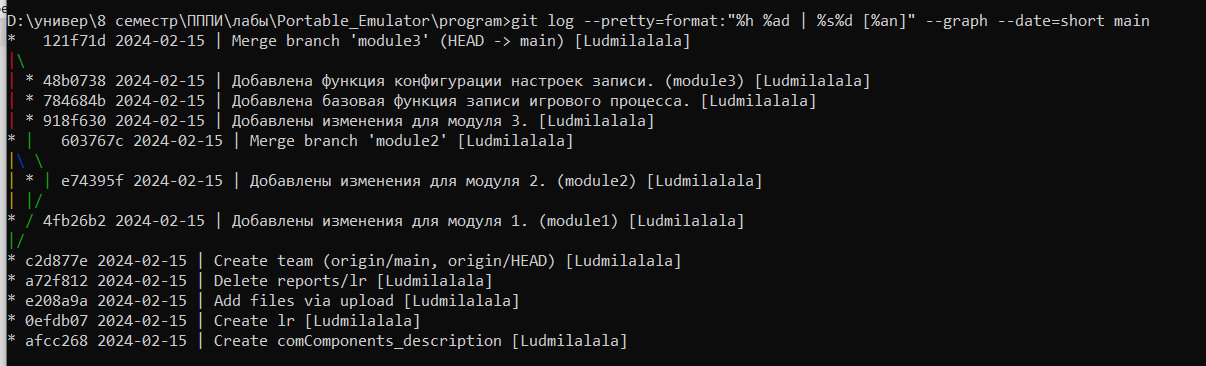


Рисунок 13 - Вывод команды git log

4 Изменить данные в одной из веток так, чтобы вызвать конфликтную ситуацию (продемонстрировать ситуацию), решить конфликтную ситуацию (также продемонстрировать).

Для демонстрации конфликтной ситуации мы можем внести изменения в одну и ту же строку в каждой из этих веток.

Чтобы продемонстрировать конфликтную ситуацию, необходимо внести изменения в одну и ту же строку в файлах, находящихся в разных ветках, а затем зафиксировать эти изменения.

Добавлен конфликтный файл hardware\_emulation.py с изменениями, которые могут вызвать конфликт при слиянии.

Внесем изменения в ветки module1 и module2 в строку *включено*. В одном случае переменная будет равна Истина, в другом Ложь.

При попытке объединить эти ветки, Git обнаружит конфликт в файле из-за различий в изменениях, внесенных в этих ветках (Рисунок 14).

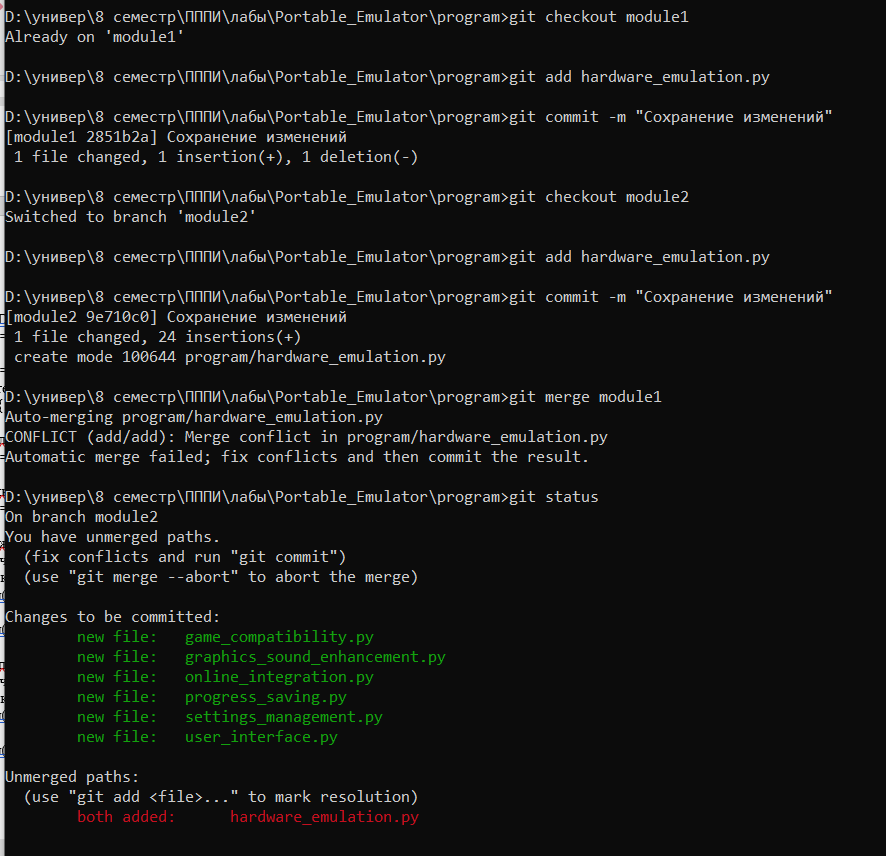


Рисунок 14 - Попытка объединить ветки

Вывод команды git status, демонстрирующую конфликтную ситуацию (Рисунок 15).

**git status -** используется для отображения текущего состояния рабочего каталога и индекса (промежуточной области) в репозитории Git

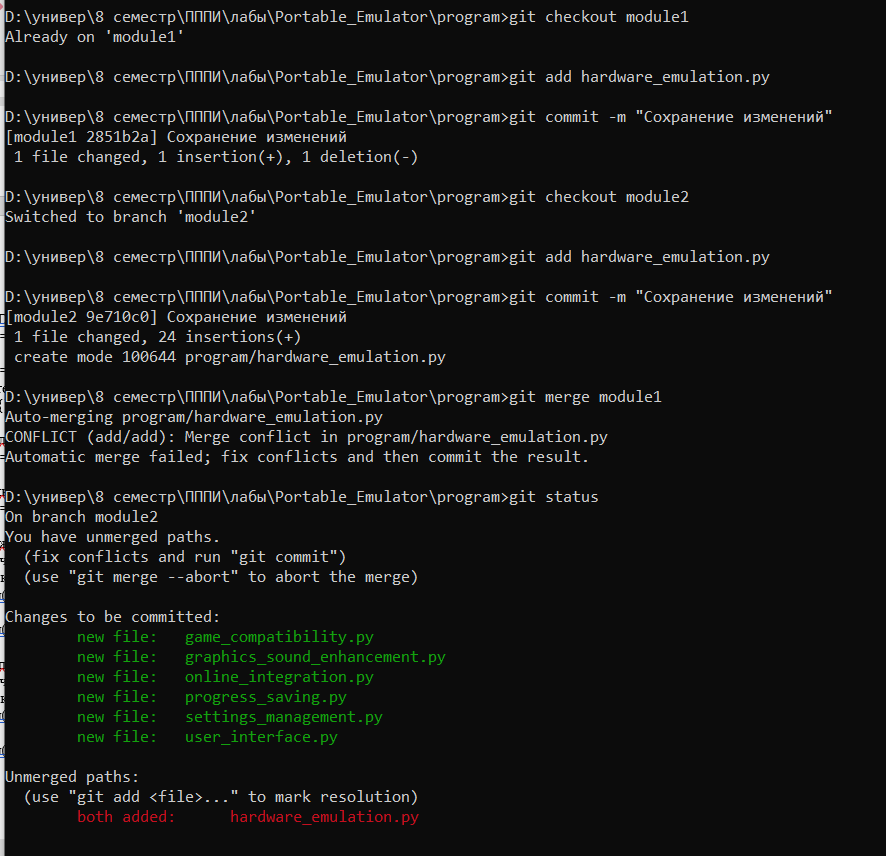


Рисунок 15 - Вывод команды git status

Содержание файла hardware\_emulation.py

hardware\_emulation.py

класс ЭмуляторОборудования:

"функция инициализации":

<<<<<<< HEAD

включено = Истина

=======

включено = Ложь

>>>>>>> module1

кнопки = {}

"функция включить\_питание":

включено = Истина

"функция выключить\_питание":

включено = Ложь

"функция нажать\_кнопку(кнопка)":

если включено:

кнопки[кнопка] = Истина

вывести(f"Кнопка {кнопка} нажата.")

иначе:

вывести("Устройство не включено.")

"функция отпустить\_кнопку(кнопка)":

если включено:

кнопки[кнопка] = Ложь

вывести(f"Кнопка {кнопка} отпущена.")

иначе:

вывести("Устройство не включено.")

Скриншот возникшего конфликта Рисунок 16

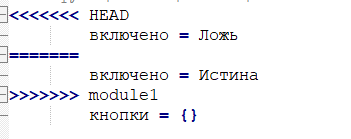


Рисунок 16 – Конфликт

Решение

Чтобы разрешить конфликт, нужно открыть файл hardware\_emulation.py

Первая секция (между <<<<<<< HEAD и =======) показывает изменения из текущей ветки (**module2**), а вторая секция (между ======= и >>>>>>> **module 1**) показывает изменения из ветки, пытаемся объединить (**module 1**).

Делаем коммит для завершения слияния

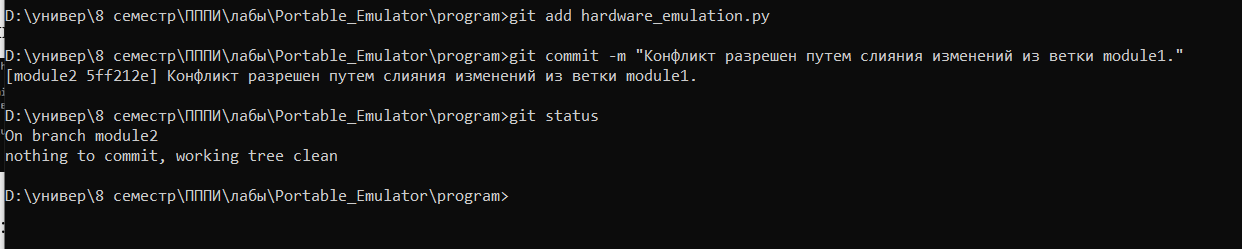


Рисунок 16 – Коммит для завершения слияния

Вывод команды git log --pretty=format:"%h %ad | %s%d [%an]" --graph --date=short (Рисунок 17).

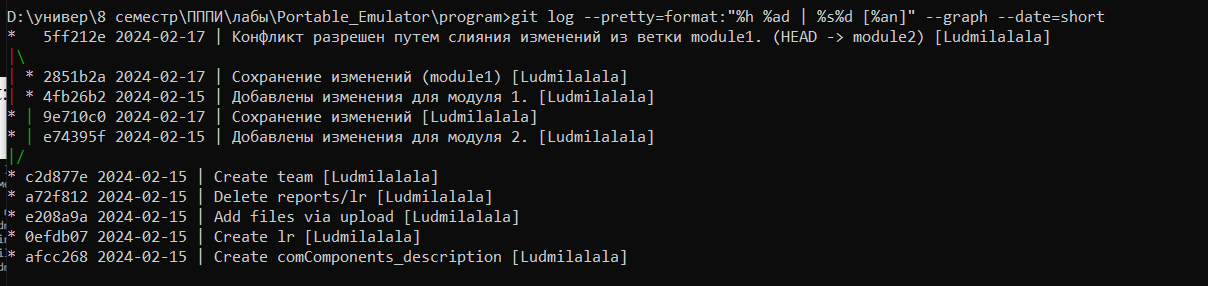


Рисунок 17 - Вывод команды git log

5 Добавить изменения в master.

Переключаемся на ветку **main.**

**git checkout main -** эта команда переключается на ветку **main** в локальном репозитории.

После выполнения слияния нет никаких конфликтов, и изменения успешно влиты в **main**, выполняем команду **git push** **origin** для отправки изменений в удаленный репозиторий (Рисунок 18):

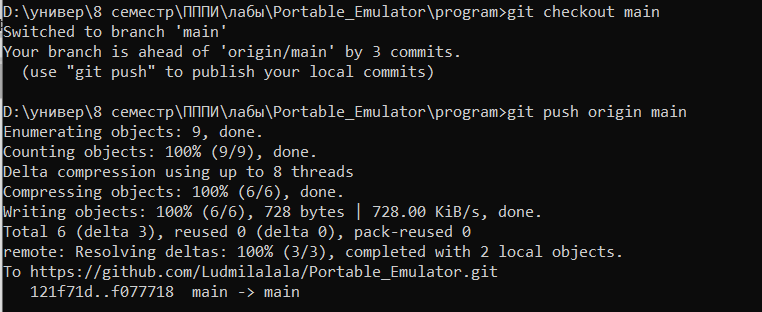


Рисунок 18 - Вывод команды git push origin

Приложение 1

**Модуль 1 (module1)**

* Эмуляция аппаратной части - hardware\_emulation.py (Рисунок 19)

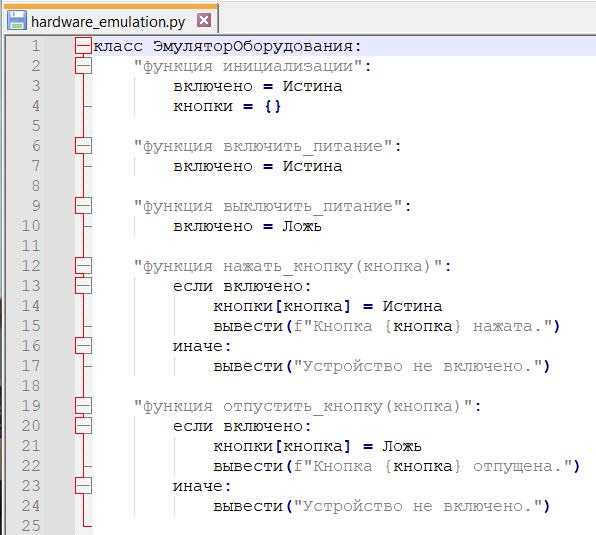


Рисунок 19 - Эмуляция аппаратной части - hardware\_emulation.py

* Графический интерфейс пользователя - user\_interface.py (Рисунок 20)

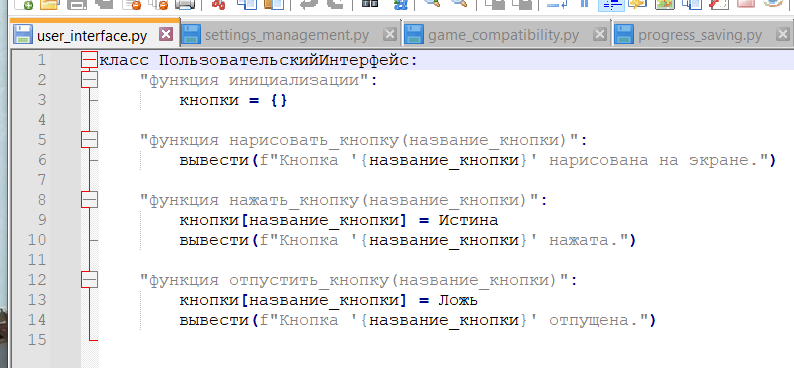


Рисунок 20 - Графический интерфейс пользователя - user\_interface.py

* Управление настройками - settings\_management.py (Рисунок 21)

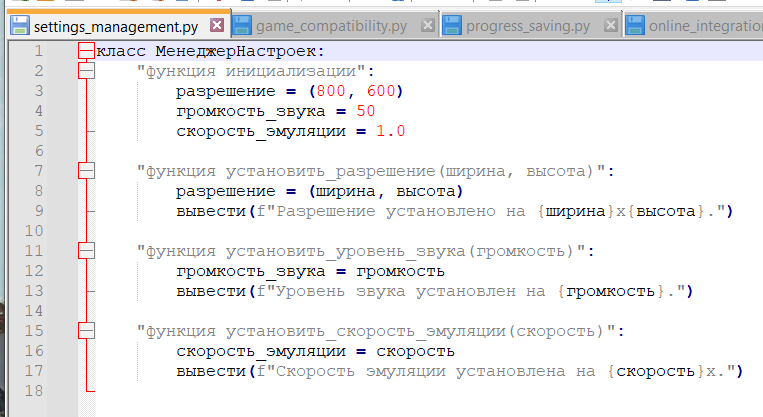


Рисунок 21 - Управление настройками - settings\_management.py

* Совместимость с играми - game\_compatibility.py (Рисунок 22)

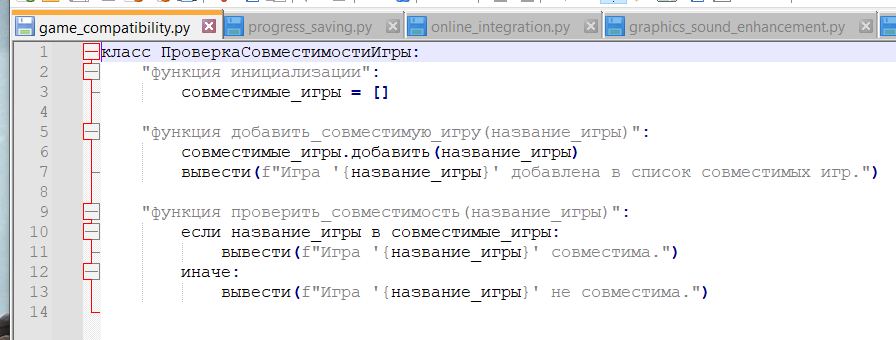
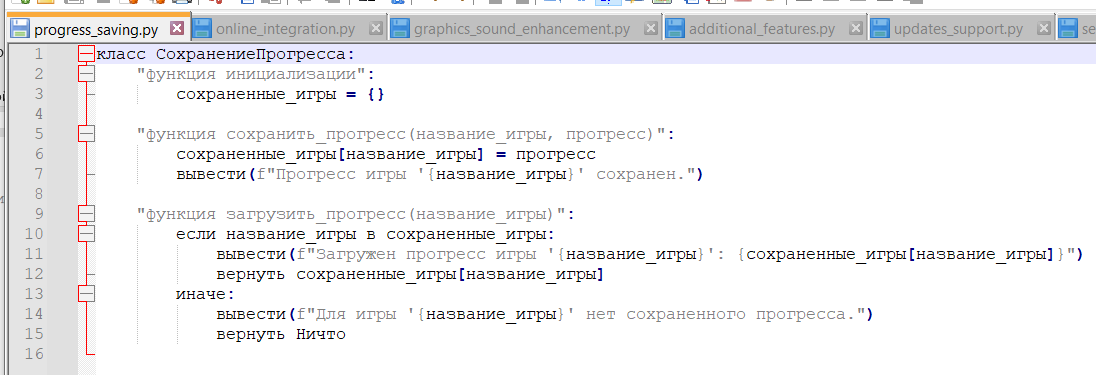


Рисунок 22 - Совместимость с играми - game\_compatibility.py

* Сохранение прогресса - progress\_saving.py (Рисунок 23)

Рисунок 23 - Сохранение прогресса - progress\_saving.py

* Интеграция с онлайн-сервисами - online\_integration.py (Рисунок 24)

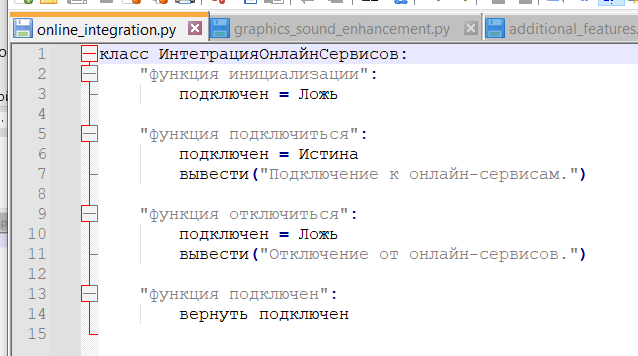


Рисунок 24 - Интеграция с онлайн-сервисами - online\_integration.py

* Улучшение графиков и звука - graphics\_sound\_enhancement.py (Рисунок 25)

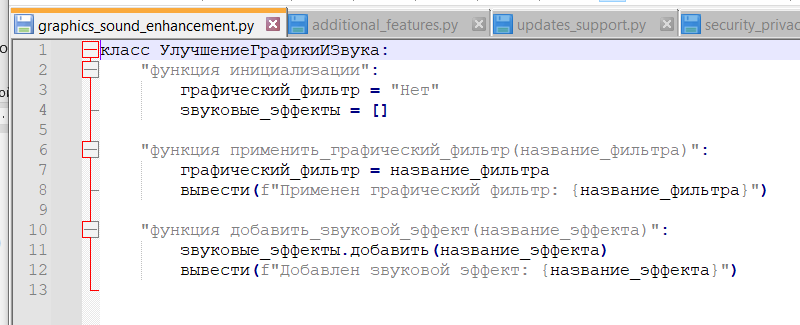


Рисунок 25 - Улучшение графиков и звука - graphics\_sound\_enhancement.py

**Модуль 2 (module2)**

* Поддержка дополнительных функций - additional\_features.py (Рисунок 26)

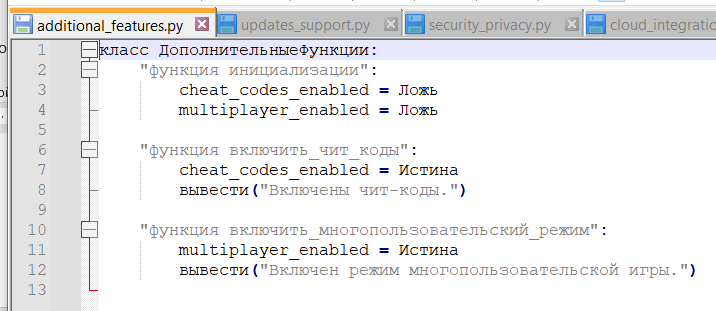


Рисунок 26 - Поддержка дополнительных функций - additional\_features.py

* Обновления и поддержка - updates\_support.py (Рисунок 27)

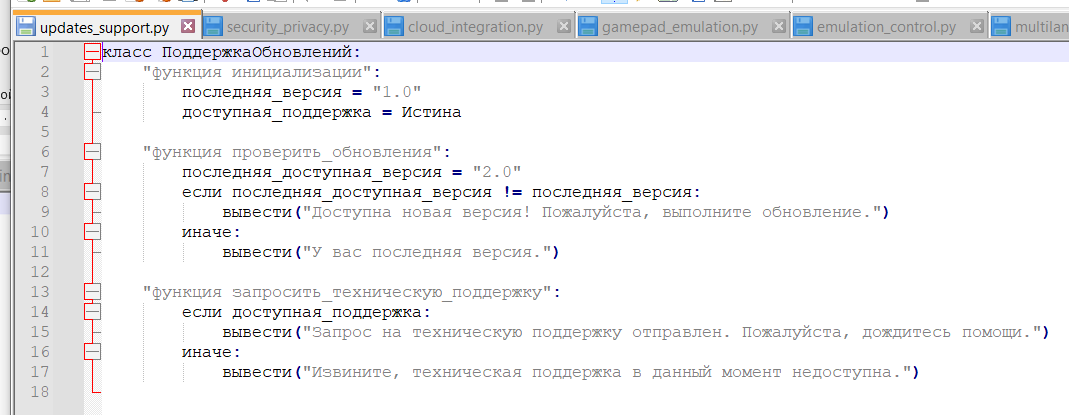


Рисунок 27 - Обновления и поддержка - updates\_support.py

* Безопасность и конфиденциальность - security\_privacy.py (Рисунок 28)

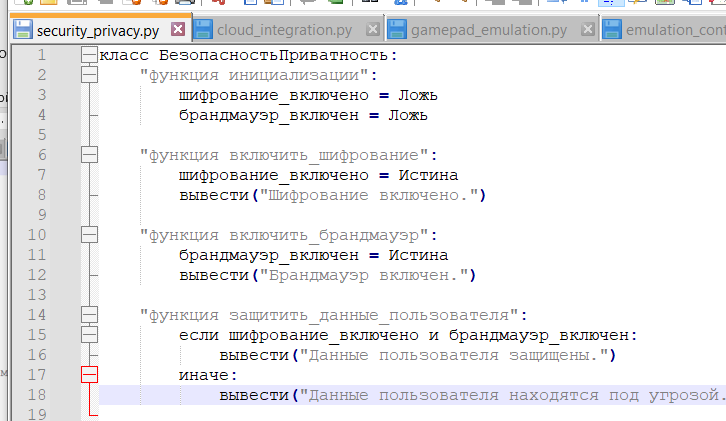


Рисунок 28 - Безопасность и конфиденциальность - security\_privacy.py

* Интеграция с облачными сервисами - cloud\_integration.py (Рисунок 29)

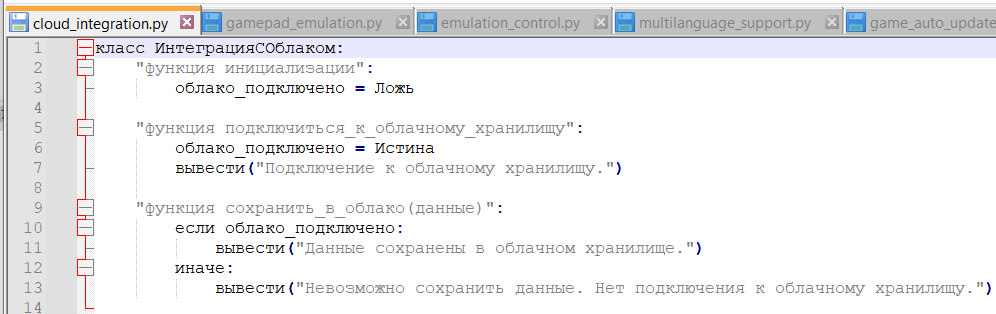


Рисунок 29 - Интеграция с облачными сервисами - cloud\_integration.py

* Геймпад-эмуляция - gamepad\_emulation.py (Рисунок 30)

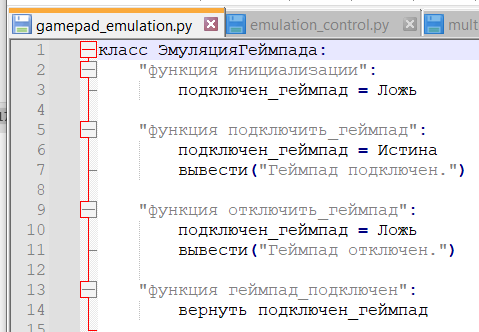


Рисунок 30 - Геймпад-эмуляция - gamepad\_emulation.py

* Модуль управления эмуляцией - emulation\_control.py (Рисунок 31)

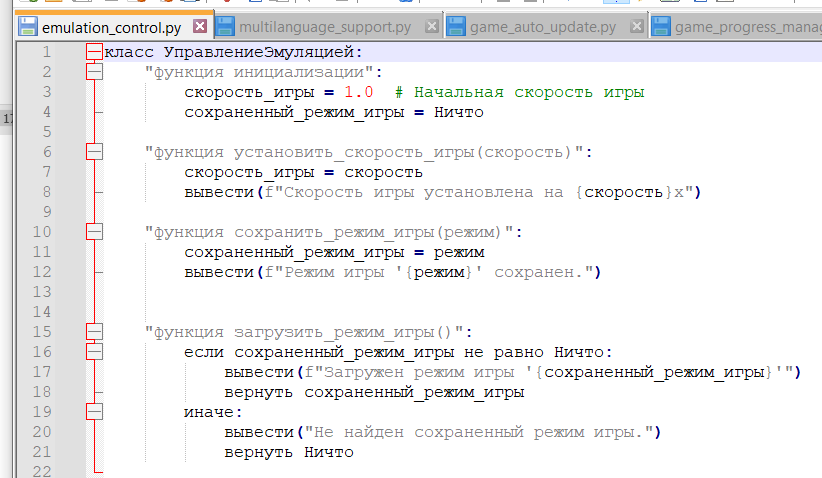


Рисунок 31 - Модуль управления эмуляцией - emulation\_control.py

* Поддержка мультиязычности - multilanguage\_support.py (Рисунок 32)

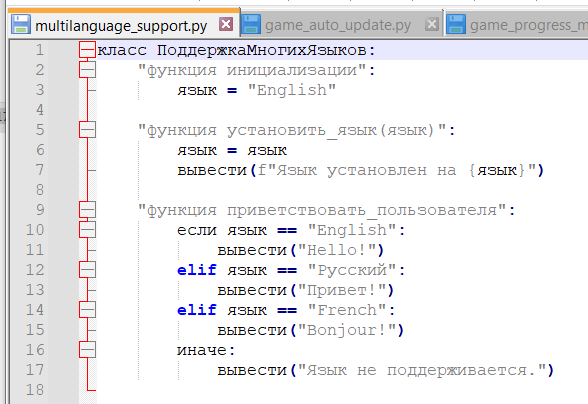


Рисунок 32 - Поддержка мультиязычности - multilanguage\_support.py

**Модуль 3 (module3)**

* Автоматическое обновление игр - game\_auto\_update.py (Рисунок 33)

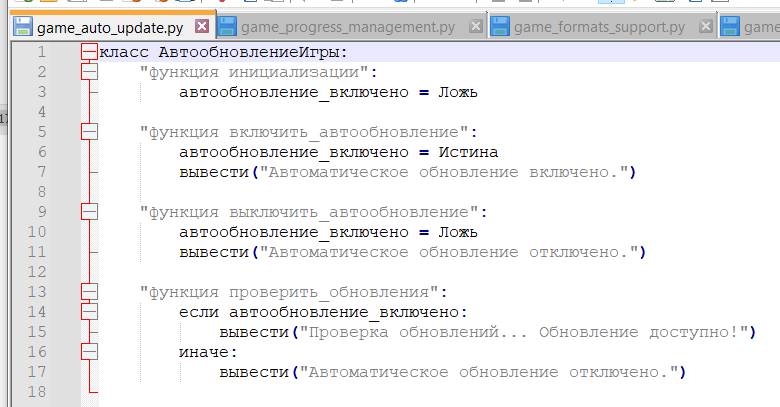


Рисунок 33 - Автоматическое обновление игр - game\_auto\_update.py

* Поддержка сохранения и загрузки процесса игры - game\_progress\_management.py (Рисунок 34)

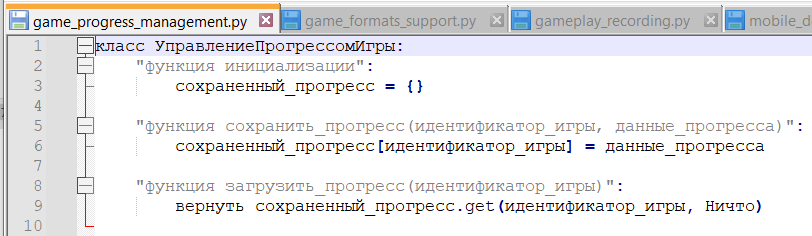


Рисунок 34 - Поддержка сохранения и загрузки процесса игры - game\_progress\_management.py

* Поддержка различных форматов игровых образов - game\_formats\_support.py (Рисунок 35)

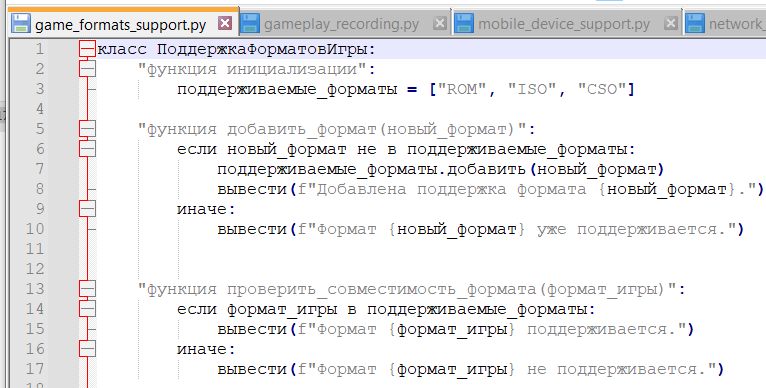


Рисунок 35 - Поддержка различных форматов игровых образов - game\_formats\_support.py

* Возможность записи игрового процесса - gameplay\_recording.py (Рисунок 36)

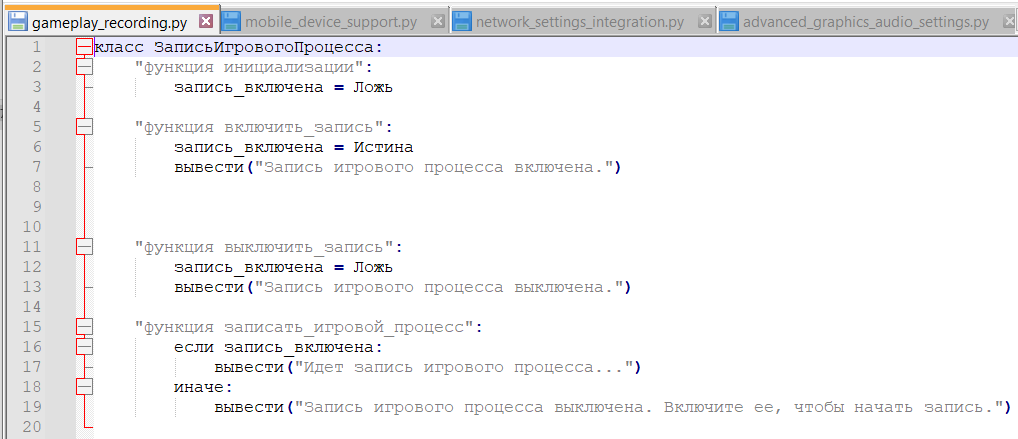


Рисунок 36 - Возможность записи игрового процесса - gameplay\_recording.py

* Поддержка мобильных устройств - mobile\_device\_support.py (Рисунок 37)

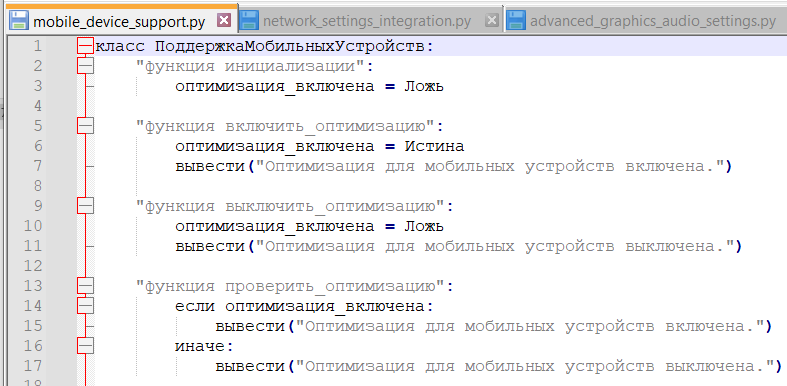


Рисунок 37 - Поддержка мобильных устройств - mobile\_device\_support.py

* Интеграция с настройками сетей - network\_settings\_integration.py (Рисунок 38)

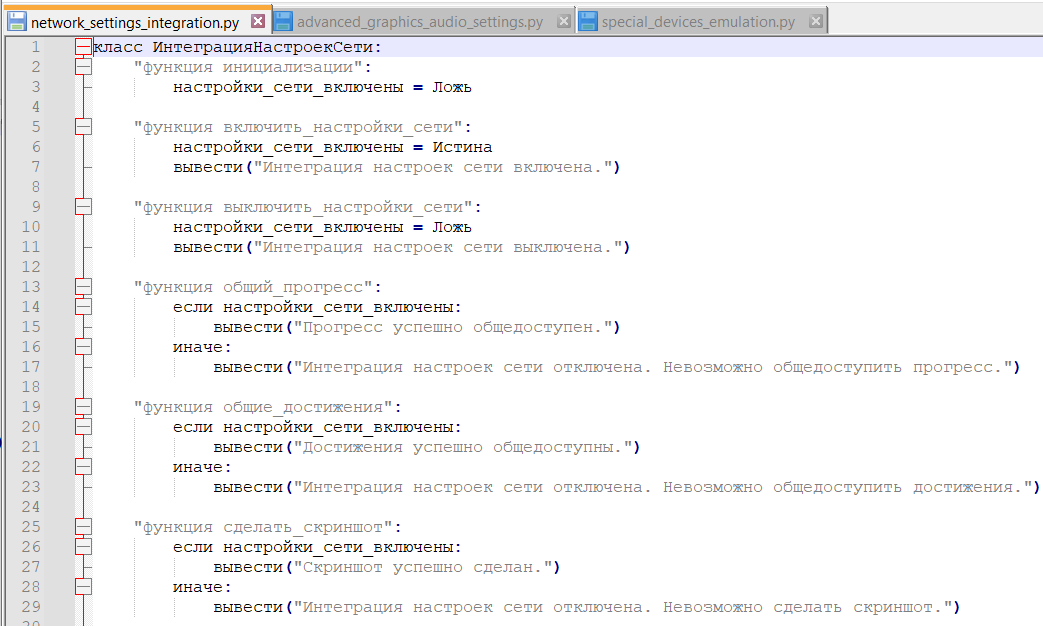


Рисунок 38 - Интеграция с настройками сетей - network\_settings\_integration.py

* Расширенные настройки графики и звука - advanced\_graphics\_audio\_settings.py (Рисунок 39)

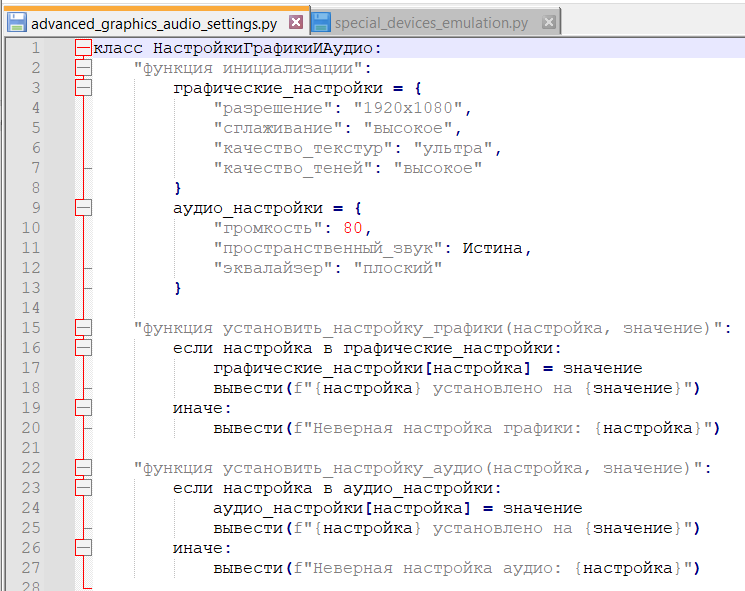


Рисунок 39 - Расширенные настройки графики и звука - advanced\_graphics\_audio\_settings.py

* Режимы эмуляции специальных устройств - special\_devices\_emulation.py (Рисунок 40)

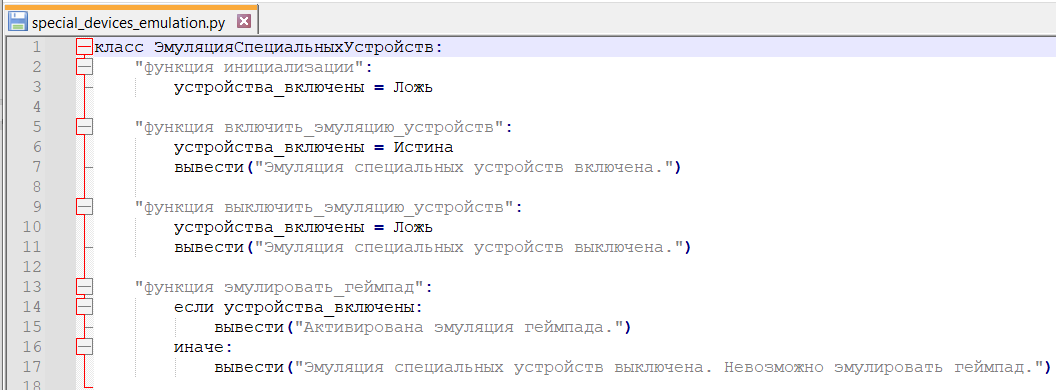


Рисунок 40 - Режимы эмуляции специальных устройств - special\_devices\_emulation.py