СУ “Св. Климент Охридски”,
ФМИ – Софтуерно инженерство
Курсов проект по Обектно-ориентирано програмиране

Game Dialog

Владислав Александрович Тимофеев, Факултетен № 855271

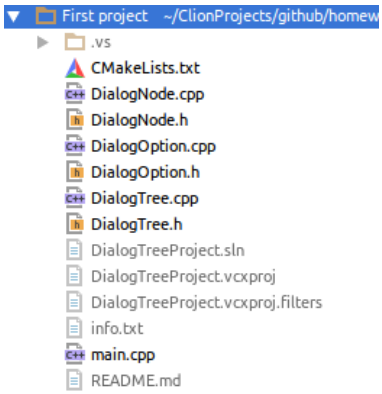
Съдържание

1. Въведение

Game Dialog – е программа, която имплементира диалози използвайки диалогово дърво.

Проектът на github: [Project (click here)]

Структура на проекта:



Readme.md — файл със допълнителни справки(за github).

.sln, .vcxproj, .vcxproj.filters — файлове за запуска чрез Visual Studio.

2. Описание на приложените алгоритми

3. Описание на програмния код

Точка на входа на програмата е файл main.cpp, в който създава се дърво myTree и започва работа на програма.

Тук променлива **way** — е път до файл със инструкциите (Как се изглежда този файл можете да прочетете във README.md).

Строка

```
//way = "/home/vtimofeev/ClionProjects/oopProject/info.txt";
```

е закоментирана за възможност въвеждане на път към файл от консола. Ако искате — можете да разкоментирате тази строка и тогава следният код (Код 1) ще разбере че това е default-value и няма да пита за въвеждане на път.

Код 1

```
if(way.length() < 2)
    getline(cin, way);
else
    cout << "Default way will be used\n";
```

Забележка. Ако използвате VS compiler – променлива **way** може да бъде като пълно име на файл с инструкциите във текуща директория. **Обаче** ако използвате cmake/other compiler – по-добре да пишете пълен път със името на файл като в примера.

Четене информация от файл е реализирана във метод void DialogTree::load(std::string way) на класс **DialogTree.** Освен отваряне на файл и проверка дали той съществува по-важен е този цикъл: (Код 3)

Код 3.

Променлива **newId** обявена извън цикъл, за да не изтрива значението ѝ при всяка итерация.

Цикъл работи до момент когато въвеждане на **newId** е грешно (това е възможно само когато EOF в нашия случай). Всяка итерация ние четем от файл Id на нов възел, след което пропускаме символа (:) до празнина, взимаме текст на възела и започваме нов вътрешен цикъл.

Вътрешен цикъл чете опциите на възела по аналогична схема, само малко корегираната. Зада четем номер на следв. Възел използваме функция **stoi()** която превежда тип „string“ към integer. Във край на всяка итерация съхраняваме новата опция във вектор **opts** и ако следващият символ е „\n“ - тогава спираме цикъл (всички опции вече са записани).

```
unsigned int newId; // reead from file

while(ifs >> newId){

    // vars for options array
    std::vector<DialogOption> opts;
    std::string newOptionText;
    int nextNode = -1;

    // vars for each node
    std::string newNodeText;

    std::string tmp;

    ifs.ignore(2, ' '); // ignoring ": "

    getline(ifs, newNodeText, '\n');

    while(getline(ifs, newOptionText, '=')){

        ifs.ignore(256, '>'); // skipping "> "

        getline(ifs, tmp, '\n'); // this thing will delete
        nextNode = stoi(tmp);

        DialogOption newOption(newOptionText, nextNode);
        opts.push_back(newOption);
        if(ifs.peek() == '\n') // if next char is '\n' - begins new node.
        {
            break;
        }
    }

    ifs.get();

    DialogNode newNode(newId, newNodeText, opts);
    nodes.push_back(newNode);
};
```

В край на основен цикъл ние създаваме нов възел като извикваме конструктор със параметри четени и вече записани преди това.

Метод void DialogTree::perfromDialog() започва диалога като започва цикъл (виж код 4)

Код 4:

```
void DialogTree::perfrormDialog() {
    std::cout << "\n[Dialog srarted]\n\n";
    int currentIndex = 0;
    do {
        std::cout << nodes[currentIndex] << std::endl;
        int countOfOptions = nodes[currentIndex].getCountOfOptions();
        int cmd; // id of option from array
        cmd = getCommand();
        while (cmd > countOfOptions || cmd == 0) {
            std::cout << "You have only " << countOfOptions << " options!" << std::endl;
            cmd = getCommand();
        }

        currentIndex = nodes[currentIndex].getOptionByIndex(cmd - 1).getNextNodeId();

    } while(currentIndex != -1);
    std::cout << "[Dialog ended]" << std::endl;
}
```

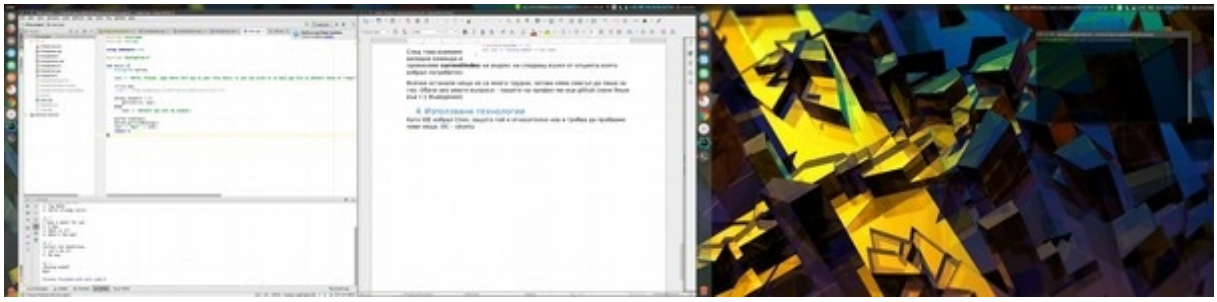
Тука **currentIndex** е индекс на текущ възел, цикъл извежда на екран информация на възел, използвайки **overloaded operator<<(**виж във DialogNode.cpp).

След това взимаме валидна команда и променяме **currentIndex** на индекс на следващ възел от опцията която избрал потребител.

Всички останали неща не са много трудни, затова няма смисъл да пиша за тях. Обаче ако имате въпроси – пишете на профил ми във github (линк беше във т.1 Въведение)

4. Използвани технологии

Като IDE избрал Clion (IDE for c++ от Jetbrains), защото той е относително нов и трябва да пробваме нови неща. ОС – ubuntu 16.04 LTS и VS 2015 &windows зада направя project-files for VS. Като технологии... малко ползвах github за маааалко по-удобна работа със 2-те системи (win+linux)



2016г., София, ФМИ