Увод

1. Основата цел на играта е Том да стигне до Джери  с дрона и по пътя евентуално да разлее цветна боя на някои места. Като самата инфрмацията за стаята се взима от файл, който потребителя може да променя. След като пусне конзолата потребителя вижда кои са най-кратките пътища от Том до Джери, и избира един от всички, като след това му се показва колко колко е дълъг и завоя и боя има по него.
2. Структура на документацията
3. Проектиране
4. Реализация, тестване
5. Реализация на играта (включва важни моменти от реализацията на класовете и малки фрагменти от кода)
6. Заключение

* Линк към репозиторито в Github:

https://github.com/ani4ka656/Tom-Jerry-SDP-Project

1. Проектиране.

Когато потребителя пусне програмата, той вижда вече дървото, съдържащо най-късите пътища, чийто върхове са инструкциите, които трябва да се въведат на дрона за стигне до Джери. Той трябва да избере един от тях като брой от ляво надясно;

Дронът на Том може да приема команди от следния тип

* **N(orth)** — лети на север
* **S(outh)** — лети на юг
* **E(ast)** — лети на изток
* **W(est)** — лети на запад
* **P(aint)** — разлей боя

Използвайки само тях, Том трябва да го “програмира”, за да реализира идеята си. Естествено има и някои ограничения:

* в стаята има мебели, над които е опасно да се лети
* батерията на дрона издържа за кратко и за това той не трябва да повтаря движенията си

За да завърши успешно мисията, котаракът трябва да вземе някои важни решения. Бързо осъзнал, че това е тежка задача, за която ще му трябва вашата помощ. Разполагате със следната информация, записана в текстов файл:

Тук е описано всичко относно структурата на самата стая, координатите, посоките, мебелите и боята.

<https://docs.google.com/document/d/1Ct_Iof1hUEWs6e4FBI57WIwbZFw-NMsAQ_JKWd2YkAk/edit>

Програмата трябва да разполага с подходящи методи, които:

* извеждат командите, които Том трябва да въведе в дрона си
* дават информация за
  + дължината на пътя
  + количеството разлята боя
  + броя завои

За да могат да се осъществят тези операции е създаден class Room, който се грижи за изпълнението им.

Член функцията start(), на класа Room по принцип трябва да изпълнява всичко.

findMinimalPaths() и printPathTo се занимават с намирането на най-късите пътища, като първата дефакто е bfs функция, която запълва масив съставен от родителите на всеки връх: (vector<vector<char> > parents), тоест ако това ни е стаята:

0 3 0

0 0 1

4 0 1

Ако Джери е 3, Том 4, мебелите са 1-ци и ако том е на позиция 6 в графа (vector<vector<pair<int, char> > > graph, във функцията g), който съдържа наредена двойка номера на съседния връх, до който маже да се стигне, и от посоките, към които може да се отиде от дадения връх. Ясно че всеки от пътищата в този случай със сигурност ще тръгнат на север към връх 3, следователно връх 3, в масива от родители ще има като родител, връх 6. В масива vector<pair<int, int> >dist съхраняваме наредена двойка от числа,. Първото показва за колко стъпки сме стигнали до даден връх, а второто колко пъти сме го достигали. В началото всеки връх е обходен -1, пъти. В примера по-горе, ако имаме ако пуснем функцията да търси пътища, като евентуално стигне джери, ще върне броя на пътищата до него, и също така dist[3],second ще върне 3, понеже толкова пъти е стигано до него, а dist[3].first ще покаже 1, понеже се намира на толкова стъпки от тази позиция; parent[3] три елемента, (n),(w), (s), зашото има път до него, идващ от тези посоки тях.

Втората трябва да извежда пътя от Том до Джери, но не работи. Поради масива от родители в нея тръгваме от Джери, за да стигнем до Том следователно пътя, който се записва е на обратно, и посоките също, следователно ги разменяме. Един път завършва, когато сме стигнали символа ‘b’, и тогава се прилага това обръщане и размяната на посоки.

Класът Tree e дървото, което трябва да запълним след като изведем пътищата(още не е реализирано.)

1. Реализация

Реализация на класове (включва важни моменти от реализацията на класовете и малки фрагменти от кода)

* class Room:

class Room

{

private:

vector<vector<int> >room;

pair<int,int> Jerry;

pair<int,int> Tom;

int furnitureCount, paintSpotsCount;

pair<int, int> paintSpots;

int n, m;

void loadGraphwithoutFurniture(vector<vector<pair<int, char> > >&);

void loadRoom(ifstream&);

int findMinimalPaths(int, int, const vector<vector<pair<int, char> > >& , vector<vector<char > >&);//bfs

void printPathTo(int v, vector<vector<char> >&parents, string = "");

public:

Room(string&);

void start();

void print();

};

* class Room:

class Room

{

private:

vector<vector<int> >room;

pair<int,int> Jerry;

pair<int,int> Tom;

int furnitureCount, paintSpotsCount;

pair<int, int> paintSpots;

int n, m;

void loadGraphwithoutFurniture(vector<vector<pair<int, char> > >&);

void loadRoom(ifstream&);

int findMinimalPaths(int, int, const vector<vector<pair<int, char> > >& , vector<vector<char > >&);//bfs

void printPathTo(int v, vector<vector<char> >&parents, string = "");

public:

Room(string&);

void start();

void print();

};

* class Tree :

class Tree

{

char data;

std::vector<Tree\*> children;

Tree\* addChild(const char c)

{

children.push\_back(new Tree(c));

return children[children.size()-1];

}

Tree\* getChild(const char c)

{

for(int i=0; i<children.size(); i++)

{

if(children[i]->data == c)

return children[i];

}

return nullptr;

}

void deleteTree()

{

for(int i=0; i<children.size(); i++)

delete children[i];

}

public:

Tree(const char data) : data(data) {}

~Tree()

{

deleteTree();

}

void addPath(std::string str)

{

if(str.size()==0)

return;

Tree\* child = getChild(str[0]);

if(child == nullptr)

child = addChild(str[0]);

child->addPath(str.substr(1));

}

void print()

{

std::cout<<data;

for(int i=0; i<children.size(); i++)

{

children[i]->print();

std::cout<<std::endl;

}

}

};

1. Заключение

* За в бъдеще трябва да се довърши дървото, да се оправят проблемите с пътищата, и да може потребителя да може да избира следователно по кой път иска да мине дронът, за да може да се извежда информацията следователно дължината на пътя, количеството разлята боя и броя завои по неговата дължина.