МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра МО ЭВМ

ОТЧЕТ

по лабораторной работе №4

по дисциплине «ООП»

Тема: Добавление класса управления игрой

Студент гр. 9383 Моисейченко К.А.

Преподаватель Жангиров Т.Р.

Санкт-Петербург
2020

Цель работы.

Создать класс игры, через который пользователь взаимодействует с игрой. А также изучить и реализовать паттерны Фасад и Команда.

Задание.

Создать класс игры, через который пользователь взаимодействует с игрой. Управление игроком, начало новой игры, завершение игры. Могут быть созданы дополнительные необходимые классы, которые отвечают отдельно за перемещение, создание игры и т.д. Но пользователь должен взаимодействовать через интерфейс одного класса.

Обязательные требования:

- Создан класс управления игрой
- Взаимодействие сохраняет инвариант

Дополнительные требования:

- Пользователь взаимодействует с использованием паттерна Команды
- Взаимодействие с компонентами происходит через паттерн Фасад

Выполнение работы.

Были созданы классы GameManager и UserInterface. Пользователь взаимодействует с игрой через класс GameManager, а отрисовка интерфейса и принятие команд реализованы в классе UserInterface.

Класс UserInterface:

Поля:

- Field* field

Методы:

+ UserInterface() - конструктор, инициализирует поле field

+ string ScanCommand() - принимает введенную пользователем команду и возвращает её

+ void PrintMenu() - выводит меню игры на экран

+ void PrintGame(Player*, int) - выводит текущее состояние игры на экран

+ void PrintHelp() - выводит справку на экран

+ void PrintRules() - выводит правила игры на экран

+ void PrintItems() - выводит информацию об игровых предметах на экран

+ void PrintControls() - выводит управление игры на экран

+ void PrintPause() - выводит окно паузы на экран

+ void PrintWin() - выводит окно победы на экран

Класс GameManager:

Поля:

- Field* field - указатель на поле игры

- Player* player - указатель на объект игрока

- UserInterface* userInterface - указатель на объект класса UserInterface

- int numOfItemPoint - число очков на поле

- int numOfItemHealth - число аптечек на поле

- int numOfItemEnergy - число джамперов на поле

- int pointsToWin - число очков, необходимых для победы

Методы:

+GameManager() - присваивает полям field и player нулевые указатели, динамически выделяет память для поля userInterface

+~GameManager() - освобождает динамически выделенную память под поля player, field, userInterface

+OpenMenu() - открывает меню и ожидает следующей команды

- std::pair<int, int> randomCell(); - возвращает координаты случайной клетки, не являющейся стеной

- void setGameObjects() - создает объекты на поле

- void openNew() - удаляет объекты, связанные с прошлой игрой, если она была, и создаёт новые объекты для начала новой игры

- void parseMove() - обрабатывает команду управления игроком

- void openHelp(int) - открывает окно справки, с помощью передаваемого в аргументе числа может вернуть пользователя на предыдущую страницу меню, в зависимости откуда открыли справку

- void openRules(int) - открывает окно правил, возвращающееся обратно в окно справки

- void openItems(int) - открывает окно информации о предметах, возвращающееся обратно в окно правил

- void openControls(int) - открывает окно управления, возвращающееся обратно в окно справки

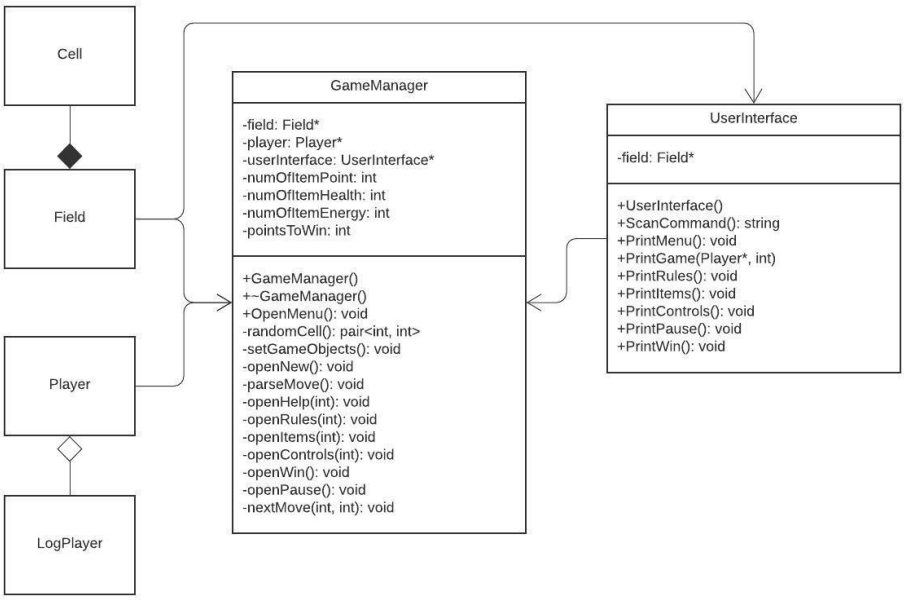
- void openWin() - открывает окно победы и ожидает команды для перехода в меню

- void openPause() - открывает окно паузы во время игры

- void nextMove(int, int) - перемещает игрока на заданные параметры, проверяет игрока на взаимодействие с элементами, проверяет игрока на условие выигрыша

Разработанный программный код см. в приложении А.

UML-диаграмма.



Выводы.

В ходе работы был создан класс игры, через который пользователь взаимодействует с игрой.

ПРИЛОЖЕНИЕ А
ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: gameManager.h

```
#pragma once

#include <stdlib.h>
#include <time.h>
#include <iostream>
#include "../Field/field.h"
#include "../Items/ItemHealth/itemHealth.h"
#include "../Items/ItemHealth/itemHealthFactory.h"
#include "../Items/ItemEnergy/itemEnergy.h"
#include "../Items/ItemEnergy/itemEnergyFactory.h"
#include "../Items/ItemPoint/itemPoint.h"
#include "../Items/ItemPoint/itemPointFactory.h"
#include "../UserInterface/userInterface.h"

class GameManager
{
private:
Field* field;
Player* player;
UserInterface* userInterface;
int numOfItemPoint;
int numOfItemHealth;
int numOfItemEnergy;
int pointsToWin;
std::pair<int, int> randomCell();
void setGameObjects();
void openNew();
void parseMove();
void openHelp(int back);
void openRules(int back);
void openItems(int back);
void openControls(int back);
void openWin();
void openPause();
void nextMove(int dx, int dy);
public:
GameManager();
void OpenMenu();
~GameManager();
};
```

Название файла: gameManager.cpp

```
#include "gameManager.h"

GameManager::GameManager()
{
field = nullptr;
player = nullptr;
userInterface = new UserInterface;
}
```

```
	std::pair<int, int> GameManager::randomCell()
	{
	int x = rand() % field->GetWidth();
	int y = rand() % field->GetHeight();

	while (field->GetField()[y][x].GetCellType() != CellType::PATH ||
field->IsItemSet(x, y))
	{
	x = rand() % field->GetWidth();
	y = rand() % field->GetHeight();
	}
	return std::make_pair(x, y);
	}

	void GameManager::setGameObjects()
	{
	ItemFactory* itemFactory;
	std::pair<int, int> randPos = randomCell();
	itemFactory = new ItemPointFactory;
	for (int i = 0; i < numOfItemPoint; i++)
	{
	randPos = randomCell();
	field->GetField()[randPos.second][randPos.first].SetItem(itemFactor
y->CreateItem());
	}
	delete itemFactory;
	itemFactory = new ItemHealthFactory;
	for (int i = 0; i < numOfItemHealth; i++)
	{
	randPos = randomCell();
	field->GetField()[randPos.second][randPos.first].SetItem(itemFactor
y->CreateItem());
	}
	delete itemFactory;
	itemFactory = new ItemEnergyFactory;
	for (int i = 0; i < numOfItemEnergy; i++)
	{
	randPos = randomCell();
	field->GetField()[randPos.second][randPos.first].SetItem(itemFactor
y->CreateItem());
	}
	delete itemFactory;
	randPos = randomCell();
	}

	void GameManager::OpenMenu()
	{
	userInterface->PrintMenu();
	std::string userCommand = userInterface->ScanCommand();
	if (userCommand == "n" || userCommand == "new")
	openNew();
	else if (userCommand == "h" || userCommand == "help")
	openHelp(0);
	else if (userCommand != "q" && userCommand != "quit")
	{
	std::cout << "Invalid command!\n";
	OpenMenu();
```

```
	}
	}

	void GameManager::openNew()
	{
	if (player)
	delete player;
	if (field)
	field->DeleteField();
	field = Field::GetInstance();
	player = new Player(field->GetStart().first,
field->GetStart().second);
	player->SetLogPlayer(new LogPlayer(player));
	srand(time(0));
	numOfItemPoint = rand() % 4 + 4;
	pointsToWin = numOfItemPoint;
	numOfItemHealth = rand() % 7 + 6;
	numOfItemEnergy = rand() % 6 + 10;
	field->SetPlayer(player);
	field->GetField()[field->GetStart().second][field->GetStart().first
].PlacePlayer(player);
	setGameObjects();
	player->GetLogPlayer()->GameStarts(pointsToWin);
	parseMove();
	}

	void GameManager::parseMove()
	{
	userInterface->PrintGame(player, numOfItemPoint);
	std::string userCommand = userInterface->ScanCommand();
	if (userCommand == "p" || userCommand == "pause")
	openPause();
	else if (userCommand == "a" || userCommand == "left")
	nextMove(-1, 0);
	else if (userCommand == "d" || userCommand == "right")
	nextMove(1, 0);
	else if (userCommand == "w" || userCommand == "up")
	nextMove(0, -1);
	else if (userCommand == "s" || userCommand == "down")
	nextMove(0, 1);
	else if (userCommand == "a2" || userCommand == "left2" || userCommand
== "2a" || userCommand == "2left")
	nextMove(-2, 0);
	else if (userCommand == "d2" || userCommand == "right2" || userCommand
== "2d" || userCommand == "2right")
	nextMove(2, 0);
	else if (userCommand == "w2" || userCommand == "up2" || userCommand
== "2w" || userCommand == "2up")
	nextMove(0, -2);
	else if (userCommand == "s2" || userCommand == "down2" || userCommand
== "2s" || userCommand == "2down")
	nextMove(0, 2);
	else if (userCommand != "q" && userCommand != "quit")
	{
	std::cout << "Invalid command!\n";
	parseMove();
	}
	}
```

```
void GameManager::openHelp(int back)
{
userInterface->PrintHelp();
std::string userCommand = userInterface->ScanCommand();
if (userCommand == "r" || userCommand == "rules")
openRules(back);
else if (userCommand == "c" || userCommand == "controls")
openControls(back);
else if (userCommand == "b" || userCommand == "back") {
if (back == 0)
OpenMenu();
else
openPause();
}
else if (userCommand != "q" && userCommand != "quit")
{
std::cout << "Invalid command!\n";
openHelp(back);
}
}

void GameManager::openRules(int back)
{
userInterface->PrintRules();
std::string userCommand = userInterface->ScanCommand();
if (userCommand == "b" || userCommand == "back")
openHelp(back);
else if (userCommand == "i" || userCommand == "items")
openItems(back);
else if (userCommand != "q" && userCommand != "quit")
{
std::cout << "Invalid command!\n";
openRules(back);
}
}

void GameManager::openItems(int back)
{
userInterface->PrintItems();
std::string userCommand = userInterface->ScanCommand();
if (userCommand == "b" || userCommand == "back")
openRules(back);
else if (userCommand != "q" && userCommand != "quit")
{
std::cout << "Invalid command!\n";
openItems(back);
}
}

void GameManager::openControls(int back)
{
userInterface->PrintControls();
std::string userCommand = userInterface->ScanCommand();
if (userCommand == "b" || userCommand == "back")
openHelp(back);
else if (userCommand != "q" && userCommand != "quit")
{
```

```
      std::cout << "Invalid command!\n";
      openControls(back);
      }
      }

      void GameManager::openWin()
      {
      userInterface->PrintWin();
      std::string userCommand = userInterface->ScanCommand();
      if (userCommand == "m" || userCommand == "menu" || userCommand ==
"<<")
      OpenMenu();
      else if (userCommand == "n" || userCommand == "new" || userCommand
== ">>")
      openNew();
      else if (userCommand != "q" && userCommand != "quit")
      {
      std::cout << "Invalid command!\n";
      openWin();
      }
      }

      void GameManager::openPause()
      {
      userInterface->PrintPause();
      std::string userCommand = userInterface->ScanCommand();
      if (userCommand == "b" || userCommand == "back")
      parseMove();
      else if (userCommand == "h" || userCommand == "help")
      openHelp(1);
      else if (userCommand == "m" || userCommand == "menu")
      OpenMenu();
      else if (userCommand != "q" && userCommand != "quit")
      {
      std::cout << "Invalid command!\n";
      openPause();
      }
      }

      void GameManager::nextMove(int dx, int dy)
      {
      if (field->IsWall(player->GetX() + dx, player->GetY() + dy)) {
      std::cout << "That's not a valid move!\n";
      parseMove();
      return;
      }
      if (abs(dx + dy) == 2)
      {
      if (player->GetEnergy() > 0)
      {
      player->UseJump();
      player->GetLogPlayer()->PlayerJumps();
      }
      else
      {
      std::cout << "You have no energy!\n";
      parseMove();
      return;
```

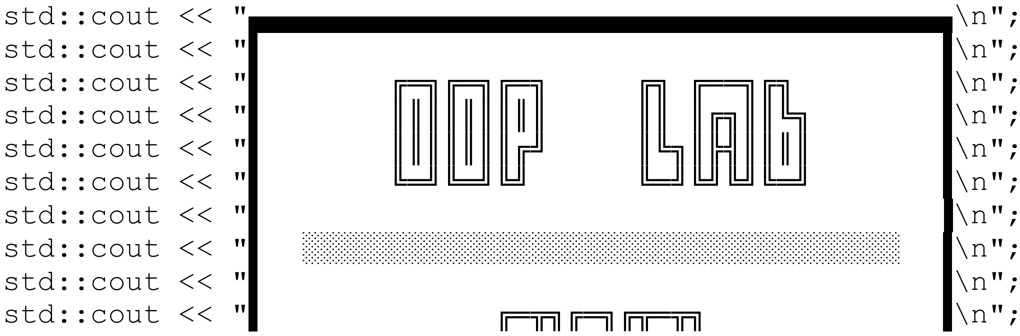
```
      }
      }
      field->GetField()[player->GetY()][player->GetX()].RemovePlayer();
      player->Move(dx, dy);
      field->GetField()[player->GetY()][player->GetX()].PlacePlayer(playe
r);
      player->GetLogPlayer()->PlayerMoves();
      if (field->IsItemSet(player->GetX(), player->GetY()))
      {
      *(field->GetItem(player->GetX(), player->GetY())) + player;
      player->GetLogPlayer()->PlayerCollects(field->GetItem(player->GetX(
), player->GetY()), pointsToWin);
      switch (field->GetItem(player->GetX(), player->GetY())->GetIndex())
      {
      case 0:
      numOfItemHealth--;
      break;
      case 1:
      numOfItemEnergy--;
      break;
      case 2:
      numOfItemPoint--;
      break;
      default:
      break;
      }
      field->GetField()[player->GetY()][player->GetX()].DeleteItem();
      }
      if  (field->GetField()[player->GetY()][player->GetX()].GetCellType()
== CellType::END)
      {
      if (player->GetPoints() == pointsToWin)
      {
      player->GetLogPlayer()->GameEnds();
      openWin();
      return;
      }
      else
      {
      std::cout << "You haven't collected all the points!\n";
      parseMove();
      return;
      }
      }
      parseMove();
      }

      GameManager::~GameManager()
      {
      if (player)
      delete player;
      if (field)
      field->DeleteField();
      if (userInterface)
      delete userInterface;
      }
```

Название файла: userInterface.h

```
#pragma once
#include <iostream>
#include <string>
#include <algorithm>
#include <cctype>
#include "../Field/field.h"

class UserInterface
{
private:
Field* field;
public:
UserInterface();
std::string ScanCommand();
void PrintMenu();
void PrintGame(Player* player, int pointsToWin);
void PrintHelp();
void PrintRules();
void PrintItems();
void PrintControls();
void PrintPause();
void PrintWin();
};
```

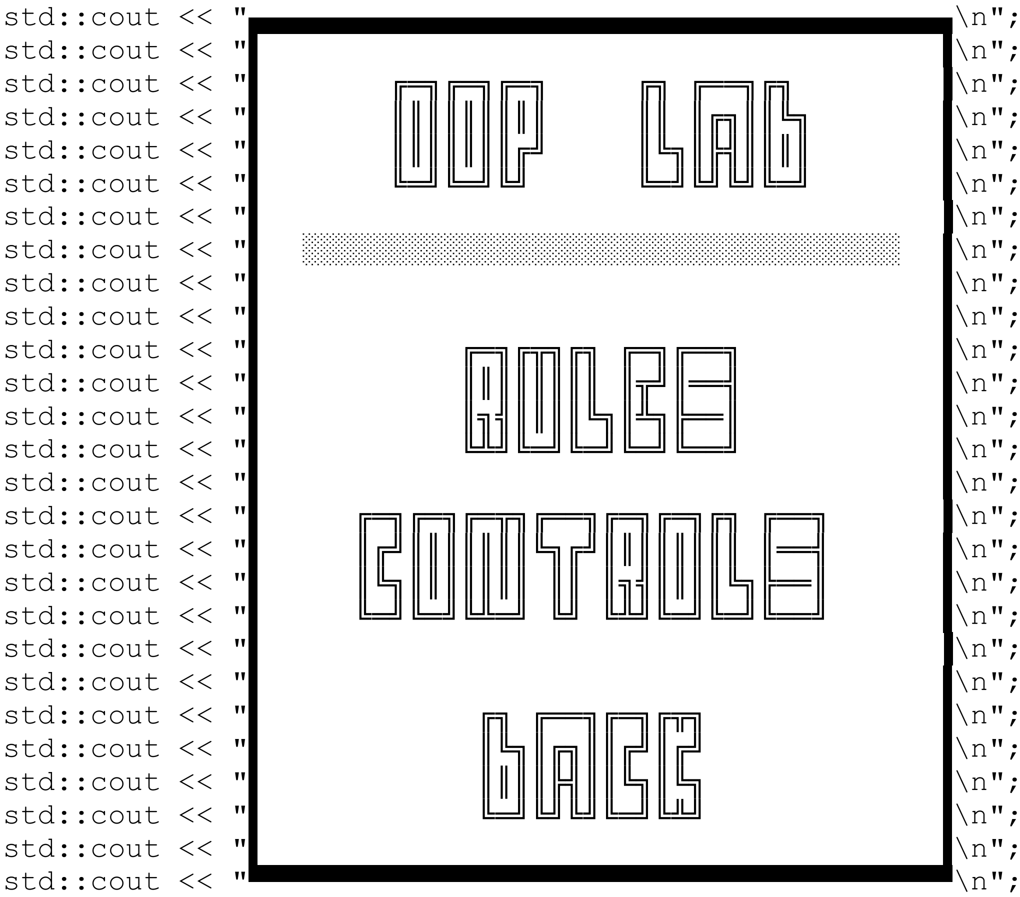
Название файла: userInterface.cpp

```
#include "userInterface.h"

UserInterface::UserInterface()
{
field = Field::GetInstance();
}

std::string UserInterface::ScanCommand() {
std::cout << "Enter your command: ";
std::string userCommand;
std::getline(std::cin, userCommand);
std::transform(userCommand.begin(),              userCommand.end(),
userCommand.begin(),
[](unsigned char c) { return std::tolower(c); });
return userCommand;
}

void UserInterface::PrintMenu()
{
```



```
      std::cout << "┃                  ╔╗╔╦╗╔╦╗                  ┃\n";
      std::cout << "┃                  ╠╣ ╚╗ ╚╗                  ┃\n";
      std::cout << "┃                  ╚╝ ═╩═ ═╩═                  ┃\n";
      std::cout << "┃                                              ┃\n";
      std::cout << "┃                                              ┃\n";
      std::cout << "┃                                              ┃\n";
      std::cout << "┃                  ╦ ╦╔═╗╦  ╔═╗                ┃\n";
      std::cout << "┃                  ╠═╣║╣ ║  ╠═╝                ┃\n";
      std::cout << "┃                  ╩ ╩╚═╝╩═╝╩                  ┃\n";
      std::cout << "┃                                              ┃\n";
      std::cout << "┃                  ╔═╗╦ ╦╦╔╦╗                  ┃\n";
      std::cout << "┃                  ║═╬╗║ ╠╣ ╚╗                 ┃\n";
      std::cout << "┃                  ╚═╝╚═╝╩ ═╩═                 ┃\n";
      std::cout << "┃                                              ┃\n";
      std::cout << "┃                                              ┃\n";
      std::cout << "┗━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━┛\n";
      }

      void UserInterface::PrintGame(Player* player, int pointsToWin) {
      std::cout << "┏━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━┓\n";
      std::cout << "┃                                              ┃\n";
      std::cout << "┃ Health<3 " << player->GetHealth();
      if (player->GetHealth() < 10)
      std::cout << "    Points{+ " << player->GetPoints() << '/' <<
pointsToWin;
      else
      std::cout << "  Points{+ " << player->GetPoints() << '/' <<
pointsToWin;
      if(player->GetEnergy() < 10)
      std::cout << "  Energy~@ " << player->GetEnergy() << " ┃\n";
      else
      std::cout << " Energy~@ " << player->GetEnergy() << " ┃\n";
      std::cout << "┃                                              ┃\n";
      for (int i = 0; i < field->GetHeight(); i++)
      {
      std::cout << "┃";
      for (int j = 0; j < field->GetWidth(); j++)
      {
      if (field->IsPlayerSet(j, i))
      std::cout << "(=";
      else if (field->IsItemSet(j, i))
      {
      switch (field->GetItem(j, i)->GetIndex())
      {
      case 0:
      std::cout << "<3";
      break;
      case 1:
      std::cout << "~@";
      break;
      case 2:
      std::cout << "{+";
      break;
      default:
      break;
      }
      }
```

```
else {
switch (field->GetField()[i][j].GetCellType())
{
case CellType::PATH:
std::cout << "  ";
break;
case CellType::WALL:
std::cout << "░░";
break;
case CellType::START:
std::cout << "->";
break;
case CellType::END:
std::cout << ">>";
break;
default:
break;
}
}
}
std::cout << " █\n";
}
std::cout << "████████████████████████████████████████\n";
}

void UserInterface::PrintHelp()
{
```

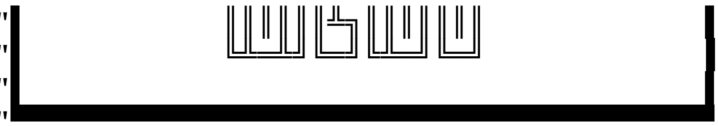


```
}

void UserInterface::PrintRules()
{
```

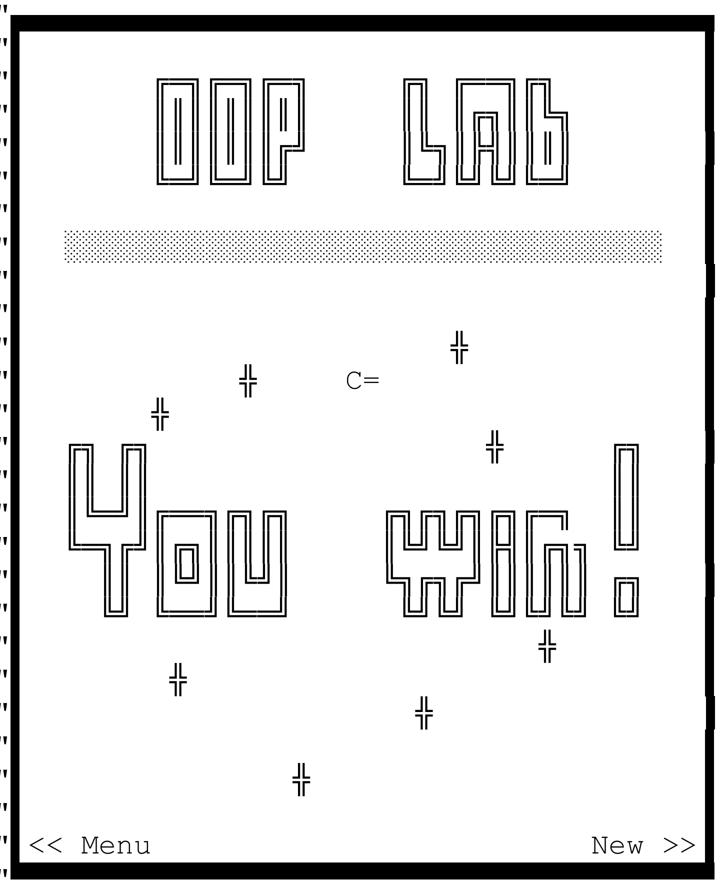
```
std::cout << "███████████████████████████████████████\n";
std::cout << "█                                     █\n";
std::cout << "█                                     █\n";
std::cout << "█                                     █\n";
std::cout << "█                                     █\n";
std::cout << "█                                     █\n";
std::cout << "█                                     █\n";
std::cout << "█                                     █\n";
std::cout << "█                                     █\n";
std::cout << "█        Collect all the points       █\n";
std::cout << "█         and reach the exit!         █\n";
std::cout << "█  -> - spawn  (= - player  >> - exit █\n";
std::cout << "█                                     █\n";
std::cout << "█                                     █\n";
std::cout << "█                                     █\n";
std::cout << "█                                     █\n";
std::cout << "█                                     █\n";
std::cout << "█                                     █\n";
std::cout << "█                                     █\n";
std::cout << "█                                     █\n";
std::cout << "█                                     █\n";
std::cout << "█                                     █\n";
std::cout << "█                                     █\n";
std::cout << "█                                     █\n";
std::cout << "█                                     █\n";
std::cout << "█                                     █\n";
std::cout << "███████████████████████████████████████\n";
}

void UserInterface::PrintItems()
{
std::cout << "███████████████████████████████████████\n";
std::cout << "█                                     █\n";
std::cout << "█                                     █\n";
std::cout << "█                                     █\n";
std::cout << "█                                     █\n";
std::cout << "█                                     █\n";
std::cout << "█                                     █\n";
std::cout << "█                                     █\n";
std::cout << "█                                     █\n";
std::cout << "█   {+ - main point, collect all      █\n";
std::cout << "█        of them to win the game      █\n";
std::cout << "█   <3 - collect to add 1 health      █\n";
std::cout << "█        to your character            █\n";
std::cout << "█   ~@ - collect to add 1 energy      █\n";
std::cout << "█        to your character            █\n";
std::cout << "█                                     █\n";
std::cout << "█    Energy allows you to jump        █\n";
std::cout << "█    over 1 cell in one move;         █\n";
std::cout << "█    you can use it to skip walls     █\n";
std::cout << "█                                     █\n";
std::cout << "█                                     █\n";
std::cout << "█                                     █\n";
std::cout << "█                                     █\n";
std::cout << "█                                     █\n";
std::cout << "█                                     █\n";
std::cout << "█                                     █\n";
std::cout << "███████████████████████████████████████\n";
```

```
}

void UserInterface::PrintControls()
{
std::cout << "\n";
std::cout << "\n";
std::cout << "\n";
std::cout << "\n";
std::cout << "\n";
std::cout << "\n";
std::cout << "\n";
std::cout << "\n";
std::cout << "\n";
std::cout << "    p/pause - pause the game\n";
std::cout << "    q/quit - quit the game\n";
std::cout << "    w/up - move up\n";
std::cout << "    a/left - move left\n";
std::cout << "    s/down - move down\n";
std::cout << "    d/right - move right\n";
std::cout << "    2 with the direction - use jump\n";
std::cout << "\n";
std::cout << "\n";
std::cout << "    All the interface commands also\n";
std::cout << "   have an abbreviated and full form\n";
std::cout << "\n";
std::cout << "\n";
std::cout << "\n";
std::cout << "\n";
std::cout << "\n";
std::cout << "\n";
std::cout << "\n";
}

void UserInterface::PrintPause()
{
std::cout << "\n";
std::cout << "\n";
std::cout << "\n";
std::cout << "\n";
std::cout << "\n";
std::cout << "\n";
std::cout << "\n";
std::cout << "\n";
std::cout << "\n";
std::cout << "\n";
std::cout << "\n";
std::cout << "\n";
std::cout << "\n";
std::cout << "\n";
std::cout << "\n";
std::cout << "\n";
std::cout << "\n";
std::cout << "\n";
std::cout << "\n";
std::cout << "\n";
std::cout << "\n";
std::cout << "\n";
std::cout << "\n";
```

```
std::cout << "                                         \n";
std::cout << "                                         \n";
std::cout << "                                         \n";
std::cout << "                                         \n";
}
```



```
void UserInterface::PrintWin()
{
std::cout << "                                         \n";
std::cout << "                                         \n";
std::cout << "                                         \n";
std::cout << "                                         \n";
std::cout << "                                         \n";
std::cout << "                                         \n";
std::cout << "                                         \n";
std::cout << "                                         \n";
std::cout << "                                         \n";
std::cout << "                                         \n";
std::cout << "                                         \n";
std::cout << "                  C=                     \n";
std::cout << "                                         \n";
std::cout << "                                         \n";
std::cout << "                                         \n";
std::cout << "                                         \n";
std::cout << "                                         \n";
std::cout << "                                         \n";
std::cout << "                                         \n";
std::cout << "                                         \n";
std::cout << "                                         \n";
std::cout << "                                         \n";
std::cout << "                                         \n";
std::cout << "                                         \n";
std::cout << "                                         \n";
std::cout << " << Menu                         New >> \n";
std::cout << "                                         \n";
}
```



Название файла: main.cpp

```
#include "../GameManager/gameManager.h"

int main() {
    GameManager* gameManager = new GameManager;
    gameManager->OpenMenu();
    if (gameManager)
        delete gameManager;
    return 0;
}
```