**МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ**

**САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ**

**ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ**

**«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)**

**Кафедра ИС**

Отчет

по курсовой работе

по дисциплине «Программирование»

Тема: «Реализация игры Pacman в консоли»

Студенты гр. 0372 Яковлев М.В.

Горбачев Н.С.

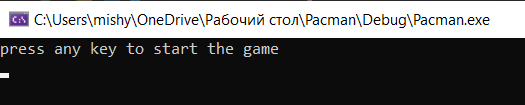
Преподаватель Квятковский А.С.

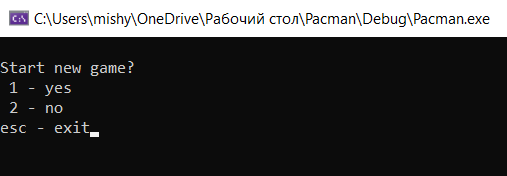
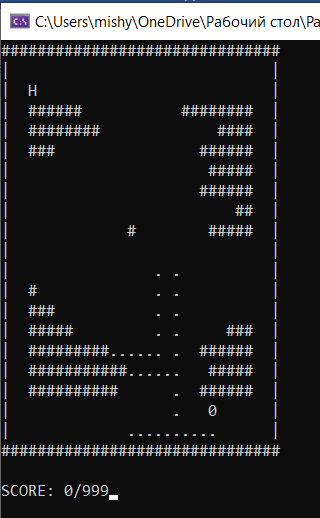
Санкт-Петербург

2020

**Цель работы**: реализовать консольную версию игры Pacman на языке программирования с++.

**Экспериментальные результаты**:



Где H – персонаж, управляемый с клавиатуры (w, a, s, d), 0 – бот, самостоятельно перемещающийся по карте.

**Протокол**:

1. Карта, в которой перемещаются персонаж и бот, реализована с помощью матрицы, которая при каждом новом запуске игры генерируется заново. Препятствия (#, |) прикрепляются к элементам матрицы и отображаются на экране. У пероснажа (Н) есть закрепленные начальные координаты спавна.

void mapmake(char\*\* map, int mapY, int mapX) {

int x(0);

for (int i = 0; i < mapY; i++) { // Массив заполняется землей

for (int j = 0; j < mapX; j++) {

map[i][j] = ' ';

}

}

for (int i = 0; i < mapY; i++) { // верхняя и нижняя граница

for (int j = 0; j < mapX; j++) {

map[i][j] = '#';

}

i += mapY - 2;

}

for (int j = 0; j < mapX; j++) { // правая и левая граница

for (int i = 1; i < mapY - 1; i++) {

map[i][j] = '|';

}

j += mapX - 2;

}

while (x < 5) {

for (int i = (mapY / 2) - 1; i < 1 + rand() % mapY; i++) { // wtynh

for (int j = (mapX / 2) - 1; j < 1 + rand() % mapX; j++) {

map[i][j] = '#';

}

}

for (int i = mapY - 4; i > rand() % mapY; i--) {//нп

for (int j = mapX - 4; j > rand() % mapX; j--) {

map[i][j] = '#';

}

}

for (int i = mapY - 4; i > rand() % mapY; i--) { // нл

for (int j = 3; j < rand() % mapX; j++) {

map[i][j] = '#';

}

}

for (int i = 3; i < rand() % mapY; i++) { // вл

for (int j = 3; j < rand() % mapX; j++) {

map[i][j] = '#';

}

}

for (int i = 3; i < rand() % mapY; i++) { // вп

for (int j = mapX - 4; j > rand() % 31; j--) {

map[i][j] = '#';

}

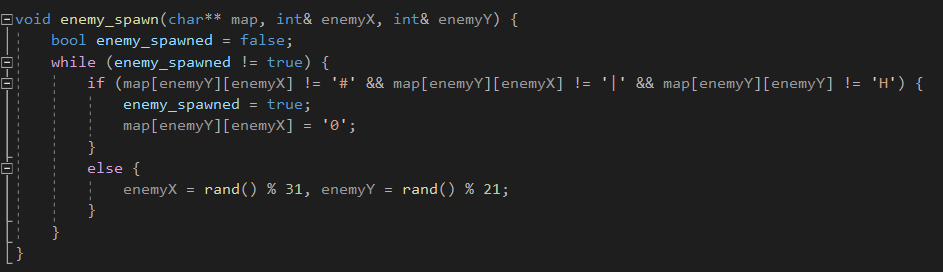
}

x++;

}

}

1. Бот спавнится в рандомной свободной позиции.



Функция спавна бота в рандомном элементе матрицы.

1. Движение бота:

void enemy\_move(char\*\* map, int& enemyX, int& enemyY, int heroY, int heroX, int score, int mapY, int mapX, int& temp) {

srand(static\_cast<unsigned int>(time(0)));

int direction;

direction = rand() % 4;

switch (direction) {

case 0: {

if (map[enemyY + 1][enemyX] == '0') {

gameover(score, temp);

map[enemyY + 1][enemyX] = ' ';

map[enemyY][enemyX] = ' ';

MapClear(map, mapY, mapX);

break;

}

else if (map[enemyY + 1][enemyX] != '#' && map[enemyY + 1][enemyX] != '|') {

enemyY++;

if (enemyY >= 0) {

map[enemyY - 1][enemyX] = '.';

map[enemyY][enemyX] = '0';

}

Sleep(100);

}

break;

}

case 1: {

if (map[enemyY - 1][enemyX] == '0') {

gameover(score, temp);

map[enemyY - 1][enemyX] = ' ';

map[enemyY][enemyX] = ' ';

MapClear(map, mapY, mapX);

break;

}

else if (map[enemyY - 1][enemyX] != '#' && map[enemyY - 1][enemyX] != '|') {

enemyY--;

if (enemyY >= 0) {

map[enemyY + 1][enemyX] = '.';

map[enemyY][enemyX] = '0';

}

Sleep(100);

}

break;

}

case 2: {

if (map[enemyY][enemyX + 1] == '0') {

gameover(score, temp);

map[enemyY][enemyX + 1] = ' ';

map[enemyY][enemyX] = ' ';

MapClear(map, mapY, mapX);

break;

}

else if (map[enemyY][enemyX + 1] != '#' && map[enemyY][enemyX + 1] != '|') {

enemyX++;

if (enemyX >= 0) {

map[enemyY][enemyX - 1] = '.';

map[enemyY][enemyX] = '0';

}

Sleep(100);

}

break;

}

case 3: {

if (map[enemyY][enemyX - 1] == '0') {

gameover(score, temp);

map[enemyY][enemyX - 1] = ' ';

map[enemyY][enemyX] = ' ';

MapClear(map, mapY, mapX);

break;

}

else if (map[enemyY][enemyX - 1] != '#' && map[enemyY][enemyX - 1] != '|') {

enemyX--;

if (enemyX >= 0) {

map[enemyY][enemyX + 1] = '.';

map[enemyY][enemyX] = '0';

}

Sleep(100);

}

break;

}

}

}

1. Движение персонажа:

if (\_kbhit()) {

key = \_getch();

switch (key) {

case 's': {// s

if (heroY == enemyY && heroX == enemyX) {

gameover(score, temp);

map[heroY][heroX] = ' ';

MapClear(map, mapY, mapX);

}

if (map[heroY + 1][heroX] == '0') {

gameover(score, temp);

map[heroY + 1][heroX] = ' ';

map[heroY][heroX] = ' ';

MapClear(map, mapY, mapX);

}

else if (map[heroY + 1][heroX] != '#' && map[heroY + 1][heroX] != '|') {

heroY++;

if (map[heroY][heroX] == '.') {

score++;

map[heroY - 1][heroX] = ' ';

map[heroY][heroX] = 'H';

}

if (heroY >= 0 && map[heroY - 1][heroX] != '0') {

map[heroY - 1][heroX] = ' ';

map[heroY][heroX] = 'H';

}

else if (map[heroY - 1][heroX] == '0') {

gameover(score, temp);

map[heroY - 1][heroX] = ' ';

map[heroY][heroX] = ' ';

MapClear(map, mapY, mapX);

}

}

break;

}

case 'w': {

if (heroY == enemyY && heroX == enemyX) {

gameover(score, temp);

map[heroY][heroX] = ' ';

MapClear(map, mapY, mapX);

}

if (map[heroY - 1][heroX] == '0') {

gameover(score, temp);

map[heroY - 1][heroX] = ' ';

map[heroY][heroX] = ' ';

MapClear(map, mapY, mapX);

}

else if (map[heroY - 1][heroX] != '#' && map[heroY - 1][heroX] != '|') {

heroY--;

if (map[heroY][heroX] == '.') {

score++;

map[heroY + 1][heroX] = ' ';

map[heroY][heroX] = 'H';

}

if (heroY >= 0 && map[heroY + 1][heroX] != '0') {

map[heroY + 1][heroX] = ' ';

map[heroY][heroX] = 'H';

}

else if (map[heroY + 1][heroX] == '0') {

gameover(score, temp);

map[heroY + 1][heroX] = ' ';

map[heroY][heroX] = ' ';

MapClear(map, mapY, mapX);

}

}

break;

}

case 'd': { //d

if (heroY == enemyY && heroX == enemyX) {

gameover(score, temp);

map[heroY][heroX] = ' ';

MapClear(map, mapY, mapX);

}

if (map[heroY][heroX + 1] == '0') {

gameover(score, temp);

map[heroY][heroX + 1] = ' ';

map[heroY][heroX] = ' ';

MapClear(map, mapY, mapX);

}

else if (map[heroY][heroX + 1] != '#' && map[heroY][heroX + 1] != '|') {

heroX++;

if (map[heroY][heroX] == '.') {

score++;

map[heroY][heroX - 1] = ' ';

map[heroY][heroX] = 'H';

}

if (heroX >= 0 && map[heroY][heroX - 1] != '0') {

map[heroY][heroX - 1] = ' ';

map[heroY][heroX] = 'H';

}

else if (map[heroY][heroX - 1] == '0') {

gameover(score, temp);

map[heroY][heroX - 1] = ' ';

map[heroY][heroX] = ' ';

MapClear(map, mapY, mapX);

}

}

break;

}

case 'a': { //a

if (heroY == enemyY && heroX == enemyX) {

gameover(score, temp);

map[heroY][heroX] = ' ';

MapClear(map, mapY, mapX);

}

if (map[heroY][heroX - 1] == '0') {

gameover(score, temp);

map[heroY][heroX - 1] = ' ';

map[heroY][heroX] = ' ';

MapClear(map, mapY, mapX);

}

else if (map[heroY][heroX - 1] != '#' && map[heroY][heroX - 1] != '|') {

heroX--;

if (map[heroY][heroX] == '.') {

score++;

map[heroY][heroX + 1] = ' ';

map[heroY][heroX] = 'H';

}

if (heroX >= 0 && map[heroY][heroX + 1] != '0') {

map[heroY][heroX + 1] = ' ';

map[heroY][heroX] = 'H';

}

else if (map[heroY][heroX + 1] == '0') {

gameover(score, temp);

map[heroY][heroX + 1] = ' ';

map[heroY][heroX] = ' ';

MapClear(map, mapY, mapX);

}

}

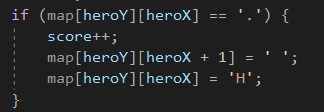
break;

}

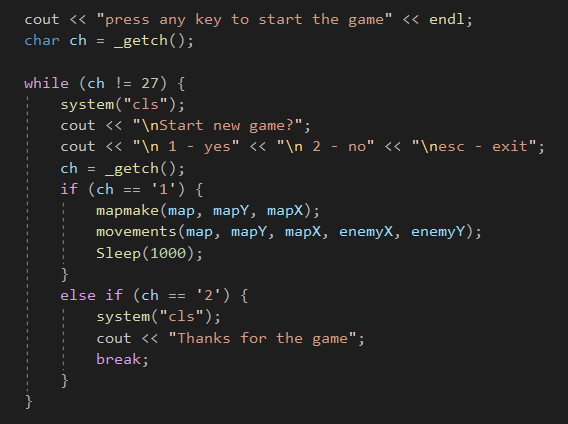
}

}

1. В нашем проекте реализован подсчет очков. После каждого своего хода бот оставляет за собой символ ‘.’. При ходе персонажа в клетку, содержащую такой символ, добавляется “SCORE”. При достижении 999 очков, игра заканчивается.

Пример досчета очков при нажании на клавишу a (eng)

1. В игре реализовано подобие меню с возможностью начала игры и с возможностью ее закрыть:

 Меню.

Выполнили: Горбачев Н.С., Яковлев М.В.

Факультет: КТИ

Группа: 0372

Преподаватель: Квятковский А.С.

Вывод: в ходе реализации проекта были закреплены основные навыки программирования на языке с++. В будущем данный проект может быть сделан в графическом виде, на основе матрицы, уже созданной в данной игре.