|  |
| --- |
| МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ |
| федеральное государственное автономное образовательное учреждение  высшего профессионального образования |
| **«Национальный исследовательский ядерный университет «МИФИ»** |



**Факультет Кибернетики и Безопасности**

**Кафедра № 36**

**Отчёт**

**По Курсовой работе**

**на тему:**

Разработка игрового приложения на C++, моделирующего процесс ведения военных действий с использованием наземной техники стратегического назначения, а именно танков.

**Студент** Рябов Петр **Группа** K3-362

**2013 г.**

***Оглавление***

1. [Постановка задачи](#Постановка_Задачи)

1.1 [*Общая задача*](#Общие)

*1.2* [*Зачем это?*](#Зачем)

*1.3* [*Частные задачи*](#Частные)

1. [Возможные решения и их анализ и выбор](#Решения)
2. [Ключевые алгоритмы](#Ключевые)   
   3.1 Общий Алгоритм Отрисовки Сцены  
    3.1.1 [Блок-Схема](#Общий_алг)   
    3.1.2 [Код](#Общий_алг_Код)   
   3.2 [Алгоритм генерации объектов](#Генерация_Об)  
    3.2.1 [Блок-Схема](#Генерация_Об)  
    3.2.2 [Код](#Генерация_Об_Код)
3. [Основные проблемы](#Основные_Проблемы)
4. [Анализ результатов](#Анализ_Результатов)
5. [Литература](#Литература)
6. [Приложения](#Прилож)

***Постановка задачи***

**-Общая Задача:** Создать приложение, моделирующее игровой процесс с использованием графики (Для GUI, Рендеринга) и поддержкой мульти плеера через сеть.

**-Зачем это?** Изучить принципы игровой разработки. Научиться использовать язык программирования C++ для создания комплексных приложений, состоящих из отдельных модулей, а также понимать принципы взаимодействия между ними . Научиться разрабатывать игровую логику и соответственно расширить свои знания по C++ и по программированию в целом.

**-Частные задачи:**

* Выбор и изучение графической и сетевой библиотеки.
* Создание графического интерфейса
* Сформировать *игровой процесс*: То есть события, которые будут происходить на игровой сцене. Возможные последствия этих событий. Внешний вид объектов и самой сцены.
* Разработать *геометрию объектов*, исходя из выбранной графической библиотеки.
* Реализовать модель взаимодействия Сервер-Клиент. Разработать оптимизированные алгоритмы передачи и обработки параметров программе клиенту от сервера.
* Продумать *систему опроса* действий пользователя, а также принципы реагирования основной программы на них.
* Определить *программную структуру,*  позволяющую отразить задуманный игровой процесс.
* Разработать *систему взаимодействия* игровых объектов, по возможности используя ООП для реализации автономных программных модулей, которые будут принимать на вход только параметры, и генерировать результат, который будет перенаправлен в основную программу.

[Вернуться к оглавлению](#Оглавление)

***Возможные решения, их анализ и выбор***

\*Примечание: выбранное решение закрашено зелёным цветом

|  |  |  |
| --- | --- | --- |
| Проблема | Решение | Анализ |
| Выбор Графической библиотеки. | **FreeGlut**  **Open GL** | Для первого раза идеально подойдет. под него множество книжных примеров. Не надо заморачиваться с инициализацией окон, получением контекста устройств зато сама логика отрисовки не очень простая. |
| **Готовый движок** | Слишком много ненужных функции в которых сложно разобраться не имея опыт. Иногда проще написать то, что нужно может не так уж хорошо, но зато понятно для себя |
| Выбор Сетевой библиотеки. | **Сеть не используется** | **На сеть не хватило времени!** Все параметры хранятся локально внутри основной программы и заголовочных файлов. |
| Геометрия объектов | **Танки в виде двух совмещённых 2D фигур: прямоугольник + пушка в виде трапеции(Полигон). Стены –прямоугольники. Пуля – отрезок** | Такую схему можно реализовать с применением графических примитивов Open GL . Эта задача более-менее реальная. |
| **3D модели + натянутые на них текстуры** | Слишком сложная математическая часть. Для текстур тоже нужно изучать отдельную библиотеку . |
| систему опроса действий пользователя | **Специальная функция обработки клавиатуры Обработка движения танка на основе прототипа, являющемся частью GLUT** *void SpecialInput(int key, int x, int y)*  **Обработка выстрела танка на основе аналогичного прототипа void Actions(unsigned char key, int x, int y) из дополнительной библиотеки GLFW** | Можно самому написать тело функции и можно можно придумать своё свой алгоритм, который будет выполняться в теле функции. Сама функция постоянно опрашивает клавиатуру по механизму CallBack.  GLFW использовалось только потому, что GLUT не умеет нормально обрабатывать пробел. Выход из игры также предусмотрен нажатием ESC |
|  | **Создание битовой маски на 256 ячеек в виде bool массива. Написать функции изменения соответствующего бита при нажатии клавиши** | GLUT не умеет перехватывать одновременное нажатие клавиш. Следовательно теряется смысл битовой маски, а сложность алгоритма возрастает |
| Создание графического интерфейса | **Простое выпадающее меню, созданное с помощью** glutCreateMenu(void (\*func)(int value)); | В любой момент можно вызвать меню правой кнопкой мыши и нажать “Play Game” и “Quit”. Лучше чем ничего. |
| Система взаимодействия | **Для объектов Пуля-Танк игрока- это генерация пули в зависимости от положения дула в момент выстрела и фиксация расстояния выстрела.** | Остальное не реализовано. |
| Программная структура | **Есть начинка из функций, которые отвечают за генерацию и обновление координат всех объектов.** | Подробно об этом написано в разделе “Ключевые алгоритмы” |

[Вернуться к оглавлению](#Оглавление)

***Ключевые алгоритмы***

1. **Общий Алгоритм Отрисовки Сцены**

Инициализация GLUT

Специальные функции работают постоянно. Если они перехватили любое из прописанных нажатие на клавишу то Основной рендеринг перезапускается

Пояснение: Пока нет выстрела пуля рендерится вторым планом под танком возле дула. В момент нажатия ПРОБЕЛ пуля вылетает и после прохожднеия дистанции объект удаляетя потом создаётся опять вторым слоем итд

Специальная функция опроса клавиатуры

Основная Функция Рендеринга

Функция Обработки изменения размеров Окна

Специальная функция опроса клавиатуры

Нет

Нажато ли “Play Game”

ДА

Отрисовка Стен

Отрисовка Пули

Отрисовка Танка Игрока

Отрисовка Ботов

**Код**

...

void Render(void) // Главная функция отрисовки игровой сцены

{

if (start)

{

glClear(GL\_COLOR\_BUFFER\_BIT); // Очистка фона

CWall(); //Рисуем стены

Shoot(); //Рисуем пулю

CTank(); //Рисуем танк

Enemy(); //Рисуем ботов

}

}

void Reshape(GLint w, GLint h) //Функция, вызывающаяся при изменени размеров окна.

//Т.е перерисовывает всё исходя из нового разрешения экрана

{

Width = w;

Height = h;

glViewport(0, 0, w, h);

glMatrixMode(GL\_PROJECTION);

glLoadIdentity();

glOrtho(0, w, 0, h, -1.0, 1.0);

glMatrixMode(GL\_MODELVIEW);

glLoadIdentity();

}

int main(int argc, char \*argv[])

{

…

glutInit(&argc, argv); //Инициализация графической библиотеки

glfwInit();

glutInitDisplayMode(GLUT\_RGB); //Режим дисплея

glutInitWindowSize(Width, Height); // Размеры окна

glClearColor(0.1, 0.1, 0.1, 0); //Задаём цвет очистики фона

glutCreateWindow("TANKS: Press Right Click to Open Game Menu"); //Окно

glutReshapeFunc(Reshape); //Обработка изменений размеров окна

glutDisplayFunc(Render); //Вызывем функцию рисования

Render();

glutSpecialFunc(SpecialInput); //вызываем опрос клавиатуры

glutKeyboardFunc(Actions); //вызываем опрос клавиатуры

glutMainLoop();

…

glfwTerminate();

return 0;

[Вернуться к оглавлению](#Оглавление)

1. **Алгоритм Генерации Объектов**

Боты Генерируются по диагонали с шагом 100

От 1-ого i до E. И от Позиции =100 до Позиция+=100

Создание экземпляра i-го объекта класса Ботов

Позиция= rand до 50 + Позиция

Создание экземпляра i-го объекта класса стен

От N.2-ого i до N

Создание экземпляра i-го объекта класса стен

От 1-ого i до N/2

Равное распределение N стен вытянутых вдоль ширины и длины достигается засчёт. Генерации случайных координат центра и фиксированной длины и ширины до N/2и после N/2 со значениями длины и ширины взятых в обратном порядке

**Код**

[Вернуться к оглавлению](#Оглавление)

void Wall\_Generator() // Генератор координат для стен

{

srand( time(0) );

for(int i=0; i<N/2; i++)

{

stena[i] = new Wall(rand()%(Width-140), rand()%(Height-140),25,75); //генерируем стенки

}

for(int i=N/2; i<N; i++)

{

stena[i] = new Wall(rand()%(Width-140), rand()%(Height-140),75,25); //генерируем стенки

}

}

void Enemy\_Spawn()

{

for(int i=0, pos=100; i<E; i++, pos+=100)

{

enemy[i]= new Tank(pos + rand() % 50, pos + rand() % 50); //Генерируем Ботов

}

}

***Основные Проблемы***

**-***Не реализована* ***проверка на коллизию*** *объектов. Танки проникают друг в друга. Со стенками аналогично.*

**-***Не получилось обработать нажатие более одной клавиши одновременно, в результате разворот и движение танка немного дёрганные. Это частично компенсируется использованием* ***Таймера*** *для задержки на малое время во время очередной отрисовкой сцены.*

**-***Нет проверки на выход за границы экрана*

**-***Игровой процесс на уровне прототипа. Есть генерация стен. Можно всего лишь двигаться и стрелять.*

**-***Никакой защиты от наслоения танка игрока на другие объекты. Стенок друг на друга во время начальной генерации.*

**-***Поворот реализован не как изменение координат, а как поворот всей камеры с предварительным сохранением в Стек* ***текущей матрицы преобразования***

*(Функции GlPushMatrix() ; координат для других объектов которые не должны разворачиваться. В последствии они восcтанавливаются функцией glPopMatrix();*

**-***Не все входные данные для отрисовки хранятся по приципу инкапсуляции. Но основные переменные и методы для объектов вынесены в отдельный заголовочный файл* ***models.h****. Данные для меню вынесены в* ***interface.h***

*Количество объектов стен ботов и дальность выстрела* ***заданы как макроопределения.***

[Вернуться к оглавлению](#Оглавление)

***Анализ Результатов***

В ходе выполнения проекта была изучена графическая библиотека FreeGlut.

А также общие принципы рендеринга OpenGL

Удалось создать псевдодинамичный игровой процесс(Есть генерация объектов, движение игрока, выстрел) который в перспективе может быть доработан.

Построена Система классов с алгоритмами обработки изменения характеристик объектов, пусть даже не полностью по принципам инкапсуляции.

Реализована пока-что однопоточная система обработки клавиш.

Не получилось создать Графический интерфейс, но есть меню игры.

[Вернуться к оглавлению](#Оглавление)

***Литература***

### *GLUT Tutorial – “Creating Menus and Submenus in GLUT”*

*Open GL “Red Book”*

*Г.Шилдт “С++ Базовый курс”*

### 

[Вернуться к оглавлению](#Оглавление)

***Приложения***

***Исходный Код главной программы***

*#define N 15 // Количество стен на карте*

*#define E 3 //Количество Ботов*

*#define BDYSH 30*

*#include <ctime>*

*#include <time.h>*

*#include <math.h>*

*#include <iostream>*

*#include <GL/glfw.h> //Подключаем заголовочный файлы графической библиотеки GLFW*

*#include <GL/glut.h> //Подключаем заголовочный файлы графической библиотеки freeglut*

*#include "models.h" //Подключаем заголовочный файл с классами объектов*

*//#include <models2.h>*

*#include "kostin.h" //Подключаем заголовочный файл с процедурами линейных преобразований координат*

*#include "interface.h"*

*using namespace std;*

*GLint Width = 1280, Height = 900; //Устанавливаем начальное разрешение экрана*

*float angle=1.1; // Переменные, отвечающие за угол поворота танка по часовой и против часовой стрелки*

*float Angle=100.0\*M\_PI/180; // Для Костина*

*float move=7.0f; // Переменная, отвечающая за сдвиг при нажатии клавиш(в пикселях)*

*//Bullet\* bul=new Bullet(bot->left+35, bot->top+35); // Создаём объект класса "Bullet";*

*Tank\* bot =new Tank(500,500);*

*Bullet\* bul =new Bullet(\*bot);*

*float xb1,xb2,xb3,xb4,yb1,yb2,yb3,yb4; //Координаты для пушки*

*int a,b=0;*

*Wall\* stena[N]; // Создаём массив указателей на стены*

*Tank\* enemy[E]; // Создаём массив указателей на ботов*

*void Shoot() //Функция отрисовки пули*

*{*

*if ((bul->bul\_y)>glutGet(GLUT\_WINDOW\_HEIGHT))*

*{*

*bul->bul\_y = bot->top+35;*

*bul->bul\_x = bot->left+35;*

*}*

*glColor3ub(195,195,0);*

*Gametimer(10);*

*glBegin(GL\_LINES);*

*{*

*glVertex2f(bul->bul\_x,bul->bul\_y+85);*

*glVertex2f(bul->bul\_x,bul->bul\_y+90);*

*}*

*glEnd();*

*}*

*void Wall\_Generator() // Генератор координат для стен*

*{*

*srand( time(0) );*

*for(int i=0; i<N/2; i++)*

*{*

*stena[i] = new Wall(rand()%(Width-140), rand()%(Height-140),25,75); //генерируем стенки*

*}*

*for(int i=N/2; i<N; i++)*

*{*

*stena[i] = new Wall(rand()%(Width-140), rand()%(Height-140),75,25); //генерируем стенки*

*}*

*}*

*void Enemy\_Spawn()*

*{*

*for(int i=0, pos=100; i<E; i++, pos+=100)*

*{*

*enemy[i]= new Tank(pos + rand() % 50, pos + rand() % 50); //Генерируем Ботов*

*}*

*}*

*void CWall() //Функция Отрисовки Стен*

*{*

*glPushMatrix();*

*glLoadIdentity();*

*glColor3ub(145,145,60);*

*for(int i=0; i<N; i++)*

*{*

*glBegin(GL\_POLYGON);*

*{*

*glVertex2f(stena[i]->x1,stena[i]->y1);*

*glVertex2f(stena[i]->x1,stena[i]->bottom);*

*glVertex2f(stena[i]->right,stena[i]->bottom);*

*glVertex2f(stena[i]->right,stena[i]->y1);*

*}*

*glEnd();*

*}*

*glPopMatrix();*

*}*

*void Enemy()*

*{*

*glPushMatrix();*

*glLoadIdentity();*

*for(int i=0; i<E; i++)*

*{*

*xb1=(enemy[i]->left+10);*

*xb2=(enemy[i]->left+30);*

*xb3=(enemy[i]->right-30);*

*xb4=(enemy[i]->right-10);*

*yb1=(enemy[i]->bottom+30);*

*yb2=(enemy[i]->top+40);*

*yb3=(enemy[i]->top+40);*

*yb4=(enemy[i]->bottom+30);*

*glBegin(GL\_POLYGON); // Блок Begin-End, отвечающий за рисование пушки (В нашем случае трапеция)*

*//Первый слой*

*{*

*glColor3ub(195,1,1); //цвет пушки танка в виде значений RGB*

*glVertex2f(xb1,yb1); //Левая нижняя точка*

*glVertex2f(xb2,yb2); //Левая верхняя точка*

*glVertex2f(xb3,yb3); //Правая верхняя точка*

*glVertex2f(xb4,yb4); // Правая нижняя точка*

*}*

*glBegin(GL\_POLYGON); // Блок Блок Begin-End,отвечающий за рисование многоугольника(В нашем случае 4 -х угольник)*

*{*

*glColor3ub(265,265,0); //цвет каркаса танка в виде значений RGB*

*glVertex2f(enemy[i]->left,enemy[i]->bottom); //Левая нижняя точка*

*glVertex2f(enemy[i]->left,enemy[i]->top); //Левая верхняя точка*

*glVertex2f(enemy[i]->right,enemy[i]->top); //Правая верхняя точка*

*glVertex2f(enemy[i]->right,enemy[i]->bottom);// Правая нижняя точка*

*}*

*glBegin(GL\_POLYGON); // Блок Begin-End, отвечающий за рисование пушки (В нашем случае трапеция)*

*//Второй слой*

*{*

*glColor3ub(265,265,2); //цвет пушки танка в виде значений RGB*

*glVertex2f(xb1,yb1); //Левая нижняя точка*

*glVertex2f(xb2,yb2); //Левая верхняя точка*

*glVertex2f(xb3,yb3); //Правая верхняя точка*

*glVertex2f(xb4,yb4); // Правая нижняя точка*

*}*

*glEnd();*

*glFinish(); //Вывод изображения*

*}*

*glPopMatrix();*

*}*

*void CTank()*

*{*

*glPushMatrix();*

*xb1=(bot->left+10);*

*xb2=(bot->left+30);*

*xb3=(bot->right-30);*

*xb4=(bot->right-10);*

*yb1=(bot->bottom+30);*

*yb2=(bot->top+40);*

*yb3=(bot->top+40);*

*yb4=(bot->bottom+30);*

*Gametimer(10);*

*glBegin(GL\_POLYGON); // Блок Begin-End, отвечающий за рисование пушки (В нашем случае трапеция)*

*//Первый слой*

*{*

*glColor3ub(195,1,1); //цвет пушки танка в виде значений RGB*

*glVertex2f(xb1,yb1); //Левая нижняя точка*

*glVertex2f(xb2,yb2); //Левая верхняя точка*

*glVertex2f(xb3,yb3); //Правая верхняя точка*

*glVertex2f(xb4,yb4); // Правая нижняя точка*

*}*

*glBegin(GL\_POLYGON); // Блок Блок Begin-End,отвечающий за рисование многоугольника(В нашем случае 4 -х угольник)*

*{*

*glColor3ub(195,195,0); //цвет каркаса танка в виде значений RGB*

*glVertex2f(bot->left,bot->bottom); //Левая нижняя точка*

*glVertex2f(bot->left,bot->top); //Левая верхняя точка*

*glVertex2f(bot->right,bot->top); //Правая верхняя точка*

*glVertex2f(bot->right,bot->bottom);// Правая нижняя точка*

*}*

*glBegin(GL\_POLYGON); // Блок Begin-End, отвечающий за рисование пушки (В нашем случае трапеция)*

*//Второй слой*

*{*

*glColor3ub(165,165,0); //цвет пушки танка в виде значений RGB*

*glVertex2f(xb1,yb1); //Левая нижняя точка*

*glVertex2f(xb2,yb2); //Левая верхняя точка*

*glVertex2f(xb3,yb3); //Правая верхняя точка*

*glVertex2f(xb4,yb4); // Правая нижняя точка*

*}*

*glEnd();*

*glFinish(); //Вывод изображения*

*glPopMatrix();*

*}*

*void Render(void) // Главная функция отрисовки игровой сцены*

*{*

*if (start)*

*{*

*glClear(GL\_COLOR\_BUFFER\_BIT); // Очистка фона*

*CWall(); //Рисуем стены*

*Shoot(); //Рисуем пулю*

*CTank(); //Рисуем танк*

*Enemy(); //Рисуем ботов*

*}*

*}*

*void Reshape(GLint w, GLint h) //Функция, вызывающаяся при изменени размеров окна.*

*//Т.е перерисовывает всё исходя из нового разрешения экрана*

*{*

*Width = w;*

*Height = h;*

*glViewport(0, 0, w, h);*

*glMatrixMode(GL\_PROJECTION);*

*glLoadIdentity();*

*glOrtho(0, w, 0, h, -1.0, 1.0);*

*glMatrixMode(GL\_MODELVIEW);*

*glLoadIdentity();*

*}*

*void Actions(unsigned char key, int x, int y)*

*{*

*if (key== GLFW\_KEY\_SPACE)*

*{*

*for(int i=0; i<BDYSH; i++)*

*{*

*bul->bul\_y += bul->vlct;*

*Render();*

*}*

*delete bul; // Реинкарнация пули опять в начало Пушки*

*Bullet\* bul =new Bullet(\*bot);*

*Render();*

*}*

*}*

*void SpecialInput(int key, int x, int y)*

*{*

*switch(key) //Опрос клавиатуры*

*{*

*case 27:*

*exit(0); // выход из приложения по нажатию клавиши ESC*

*break;*

*case GLUT\_KEY\_RIGHT:*

*glTranslatef((bot->left)+bot->lgth, (bot->bottom)+bot->wdth, 0);*

*glRotatef(-angle,0.0,0.0,1.0);*

*glTranslatef(((bot->left)+bot->lgth)\*(-1), ((bot->bottom)+bot->wdth)\*(-1), 0);*

*bot->turntorght();*

*cout<<bot->azimut<<endl;*

*break;*

*case GLUT\_KEY\_LEFT:*

*glTranslatef((bot->left)+bot->lgth, (bot->bottom)+bot->wdth, 0);*

*glRotatef(angle,0.0,0.0,1.0);*

*glTranslatef(((bot->left)+bot->lgth)\*(-1), ((bot->bottom)+bot->wdth)\*(-1), 0);*

*bot->turntolft();*

*cout<<bot->azimut<<endl;*

*break;*

*case GLUT\_KEY\_UP: //Вверх*

*//Kostin(bot->left,bot->bottom,7);*

*//Kostin(bot->left,bot->top,7);*

*//Kostin(bot->right,bot->top,7);*

*//Kostin(bot->right,bot->bottom,7);*

*bot->go\_frwd();*

*bul->bul\_y+=1.0;*

*cout<<bot->centre\_x<<" "<<bot->centre\_y<<endl;*

*break;*

*case GLUT\_KEY\_DOWN: //Вниз*

*//Kostin(bot->left,bot->bottom,8);*

*//Kostin(bot->left,bot->top,8);*

*//Kostin(bot->right,bot->top,8);*

*//Kostin(bot->right,bot->bottom,8);*

*bot->go\_back();*

*bul->bul\_y-=1.0;*

*cout<<bot->centre\_x<<" "<<bot->centre\_y<<endl;*

*break;*

*}*

*glutPostRedisplay();*

*}*

*int main(int argc, char \*argv[])*

*{*

*Wall\_Generator();*

*Enemy\_Spawn();*

*glutInit(&argc, argv); //Инициализация графической библиотеки*

*glfwInit();*

*glutInitDisplayMode(GLUT\_RGB); //Режим дисплея*

*glutInitWindowSize(Width, Height); // Размеры окна*

*glClearColor(0.1, 0.1, 0.1, 0); //Задаём цвет очистики фона*

*glutCreateWindow("TANKS: Press Right Click to Open Game Menu"); //Окно*

*glutReshapeFunc(Reshape); //Обработка изменений размеров окна*

*createMenu();*

*glutDisplayFunc(Render); //Вызывем функцию рисования*

*Render();*

*glutSpecialFunc(SpecialInput); //вызываем опрос клавиатуры*

*glutKeyboardFunc(Actions); //вызываем опрос клавиатуры*

*glutMainLoop();*

*delete bot; //Удаляем танк*

*delete bul; //Удаляем пули*

*for(int i=0; i<N; i++) // Удаляем стены*

*{*

*delete stena[i];*

*}*

*for(int i=0; i<E; i++) // Удаляем стены*

*{*

*delete enemy[i];*

*}*

*glfwTerminate();*

*return 0;*

*}*

***Исходный Код models.h***

[Вернуться к оглавлению](#Оглавление)

***Исходный Код models.h***

void Gametimer(int ms) {

int CLOCKS\_PER\_MSEC = CLOCKS\_PER\_SEC / 1000; // новая константа

clock\_t end\_time = clock() + ms \* CLOCKS\_PER\_MSEC ; // время завершения

while (clock() < end\_time) {} // цикл ожидания времени

}

class Tank{

public:

const double lgth=35;

const double wdth=50;

double centre\_x; //координаты центра танка

double centre\_y;

int azimut; //угол от направления наверх, куда смотрит дуло

float left, right, bottom, top; //опять же таки дубль

Tank(double x, double y){ //конструктор

azimut=0;

centre\_x=x;

centre\_y=y;

left= x-lgth; right= x+lgth; bottom= y-wdth; top= y+wdth;

}

void go\_frwd(){ //идти вперед

azimut=0;

centre\_x+=sin(azimut\*M\_PI/180);

centre\_y+=cos(azimut\*M\_PI/180);

left= centre\_x-lgth; right= centre\_x+lgth; //переприсвоение для отрисовки, как я понимаю

bottom= centre\_y-wdth; top= centre\_y+wdth;

}

void go\_back(){ //аналогично назад

azimut=0;

centre\_x-=sin(azimut);

centre\_y-=cos(azimut);

left= centre\_x-lgth; right= centre\_x+lgth; bottom= centre\_y-wdth; top= centre\_y+wdth;

}

void turntorght(){ //поворот направо, на 1 градус

//azimut++;

}

void turntolft(){ //аналогично налево

//azimut--;

}

};

class Bullet{

public:

double bul\_x, bul\_y;

int azimut;

const int vlct=5; //скорость полета пули при выстреле

Bullet(Tank my\_T){ //конструктор, на вход получает танк, которому принадлежит

azimut=my\_T.azimut;

bul\_x=my\_T.centre\_x; //+50\*sin(azimut);

bul\_y=my\_T.centre\_y; //+90\*cos(azimut);

}

void fly(double str\_time){ //полет пулиGLUT\_KEY\_UP, на вход начальное время

while(bul\_x<1280 && bul\_y<900){

bul\_x+=vlct\*difftime(str\_time,clock())/CLOCKS\_PER\_SEC;

bul\_y+=vlct\*difftime(str\_time,clock())/CLOCKS\_PER\_SEC;

}

}

};

class Wall

{

public:

float x1,y1;

float x, y;

float left, right, bottom, top; //x1,y1 - коордианаты верхнего левого угла, length - ширина, height - высота. При создании объекта конструктор принимает на вход координаты, высота, ширину. Создаётся стена с координатами верхнего левого угла в этой точке, заданной шириной и высотой.

Wall(float in\_x, float in\_y, float length, float heigth): x1(in\_x),y1(in\_y), right(in\_x+length), bottom(in\_y+heigth), top(in\_y)

{

}

};

***Исходный Код interface.h***

*#include <GL/glut.h> //Подключаем заголовочный файлы графической библиотеки freeglut*

*static int window;*

*static int menu\_id;*

*bool start=false; //Логическая переменная отвечающая за пуск игры*

*void menu(int num)*

*{*

*if(num == 0) //Если нажали Quit*

*{*

*glutDestroyWindow(window);*

*exit(0);*

*}*

*if(num == 1) //Есди нажали Play Game*

*{*

*start=true;*

*//glutRemoveMenuItem(1); // Удаляем элемент Play Game*

*glutPostRedisplay();*

*}*

*}*

*void createMenu(void) //Создаём меню*

*{*

*glClearColor(0.1, 0.1, 0.1, 0); //Задаём цвет очистики фона*

*glClear(GL\_COLOR\_BUFFER\_BIT); // Очистка фона*

*menu\_id = glutCreateMenu(menu);*

*glutAddMenuEntry(" Play Game ", 1);*

*glutAddMenuEntry(" Quit ", 0);*

*glutAttachMenu(GLUT\_RIGHT\_BUTTON);*

*}*