```cpp
#pragma once
#include<iostream>

using namespace std;

class Point
{
protected:
    int x;
    int y;
public:
    int GetX() { return x; }
    void SetX(int xn) { x = xn; }
    int GetY() { return y; }
    void SetY(int yn) { y = yn; }

    Point();
    Point(int xn, int yn);
    Point(const Point &a);

    friend istream &operator >> (istream &in, Point &c);  //ввод из файла
};

class BoxCollider
{
private:
    Point DLP;  //Нижняя левая точка
    Point URP;  //Правая нижняя точка
public:
    BoxCollider();
    BoxCollider(int x1, int y1, int x2, int y2);
    BoxCollider(Point a1, Point a2);
    BoxCollider(const BoxCollider &b);
    ~BoxCollider() {}

    Point GetDLP() {return DLP;}
    Point GetURP(){return URP;}

    void SetBoxCollider(Point a1, Point a2);

    bool operator && (BoxCollider &c);      //Логическая операция пересечения двух коллайдеров
    friend istream &operator >> (istream &in, BoxCollider &c);  //ввод из файла
};
```

```cpp
#include "BoxCollider.h"
#include <fstream>

//Методы класса Point
Point::Point()
{
    x = 0;
    y = 0;
}

Point::Point(int xn, int yn)
{
    x = xn;
    y = yn;
}

Point::Point(const Point &a)
{
    x = a.x;
    y = a.y;
}


istream &operator >> (istream &in, Point &c)
{
    in >> c.x >> c.y;
    return in;
}

//Методы класса BoxCollider
BoxCollider::BoxCollider()
{
    DLP.SetX(0);
    DLP.SetY(0);
    URP.SetX(0);
    URP.SetY(0);
}

BoxCollider::BoxCollider(int x1, int y1, int x2, int y2)
{
    DLP.SetX(x1);
    DLP.SetY(y1);
    URP.SetX(x2);
    URP.SetY(y2);
}

BoxCollider::BoxCollider(Point a1, Point a2)
{
    DLP = a1;
    URP = a2;
}

void BoxCollider::SetBoxCollider(Point a1, Point a2)
{
    DLP = a1;
    URP = a2;
}

bool BoxCollider::operator && (BoxCollider &c)
{
    return (URP.GetY() >= c.GetDLP().GetY()) && (DLP.GetY() <= c.GetURP().GetY()) && (URP.GetX() >=
      c.GetDLP().GetX()) && (DLP.GetX() <= c.GetURP().GetX());
}

```

```
64  BoxCollider::BoxCollider(const BoxCollider &b)
65  {
66      DLP = b.DLP;
67      URP = b.URP;
68  }
69
70  istream &operator >> (istream &in, BoxCollider &c)
71  {
72      in >> c.DLP >> c.URP;
73      return in;
74  }
```

```h
#pragma once
#include "BoxCollider.h"
#include <fstream>

class Bullet :public Point
{
protected:
    int speed;
    int p;      //направление движения 1-вверх, 2-право, 3-влево, 4-вниз
    bool enemy; //вражеская пуля или нет
public:
    BoxCollider Collider;

    Bullet();
    Bullet(int xn, int yn, int s, int pn, BoxCollider bc, bool en);
    Bullet(const Bullet &a);

    int GetSpeed() { return speed; }
    void SetSpeed(int s) { speed = s; }
    int GetP() { return p; }
    void SetP(int pn) { p = pn; }
    bool GetEnemy() { return enemy; }
    void SetEnemy(bool en) { enemy = en; }

    void MoveBullet();  //движение пули

    friend istream &operator >> (istream &in, Bullet &c); //ввод из файла
};

class Wall :public Point
{
private:
    int type;   //тип стены 1-непробивая, 2-кирпичная, 3-вода
public:
    BoxCollider WallCollider;

    Wall();
    Wall(int xn, int yn, int t, BoxCollider bc);
    Wall(const Wall &a);

    int GetType() { return type; }
    void SetType(int t) { type = t; }

    friend istream &operator >> (istream &in, Wall &c); //ввод из файла
};
```

```cpp
#include "Bullet.h"

//Методы класса Bullet
Bullet::Bullet() :Point()
{
    speed = 0;
    p = 0;
    enemy = false;
}

Bullet::Bullet(int xn, int yn, int s, int pn, BoxCollider bc, bool en) :Point(xn, yn)
{
    speed = s;
    p = pn;
    Collider = bc;
    enemy = en;
}

Bullet::Bullet(const Bullet &a) :Point(a)
{
    speed = a.speed;
    p = a.p;
    Collider = a.Collider;
    enemy = a.enemy;
}

void Bullet::MoveBullet()
{
    switch (p)
    {
    case 1:
        this->y = this->y + speed;
        break;
    case 2:
        this->x = this->x + speed;
        break;
    case 3:
        this->x = this->x - speed;
        break;
    case 4:
        this->y = this->y - speed;
        break;
    }
    Collider.SetBoxCollider(Point(this->x - 2, this->y - 2), Point(this->x + 2, this->y + 2));
}

istream &operator >> (istream &in, Bullet &c)
{
    in >> (Point)c >> c.Collider >> c.speed >> c.p >> c.enemy;
    return in;
}

//Методы класса Wall
Wall::Wall() :Point()
{
    type = 0;
}

Wall::Wall(int xn, int yn,int t, BoxCollider bc) :Point(xn, yn)
{
    type = t;
    WallCollider = bc;
}

```

```
65  Wall::Wall(const Wall &a):Point(a)
66  {
67      type = a.type;
68      WallCollider = a.WallCollider;
69  }
70
71  istream &operator >> (istream &in, Wall &c)
72  {
73      in >> c.x >> c.y;
74      in >> c.WallCollider >> c.type;
75      return in;
76  }
```

```
#pragma once
#include <vector>
#include <ctime>
#include<Windows.h>
#include"BoxCollider.h"
#include"Bullet.h"
#include"Control.h"

class Tank :public Bullet
{
private:
    int hp;
    bool attack;
    bool target=false;
    void ChangeCol();
    bool CheckMoveAi(vector<Wall> &w, vector<Tank> &a, Tank &t);
    bool CheckMove(vector<Wall> &w, vector<Tank> &a);
    void MoveUpAi(vector<Wall> &w, vector<Tank> &a, Tank &t);
    void MoveRightAi(vector<Wall> &w, vector<Tank> &a, Tank &t);
    void MoveLeftAi(vector<Wall> &w, vector<Tank> &a, Tank &t);
    void MoveDownAi(vector<Wall> &w, vector<Tank> &a, Tank &t);
    void MoveUp(vector<Wall> &w, vector<Tank> &a);
    void MoveRight(vector<Wall> &w, vector<Tank> &a);
    void MoveLeft(vector<Wall> &w, vector<Tank> &a);
    void MoveDown(vector<Wall> &w, vector<Tank> &a);

public:
    Tank();
    Tank(int xn, int yn, int hpn, int s, int pn, bool at, bool en, BoxCollider bc);
    Tank(const Tank &a);

    int GetHp() { return hp; }
    void SetHp(int h) { hp = h; }
    bool GetAttack() { return attack; }
    void SetAttack(bool at) { attack = at; }
    bool GetTarget() { return target; }
    void SetTarget(bool t) { target = t; }

    void Shoot(int Cooldown, vector<Bullet> &b, Tank &a);
    void MoveAi(vector<Wall> &w, vector<Tank> &tI, Tank &t);
    void MovePlayer(Controls k, vector<Wall> &w, vector<Tank> &tI);
    void FindTarget(Tank pl, Tank &ii);

    friend istream &operator >> (istream &in, Tank &c);
    friend void ShootCooldown(int Cooldown,vector<Bullet> &b, Tank &a);
    friend void FT(Tank pl, Tank &ii);
};
```

```cpp
#define _CRT_SECURE_NO_WARNINGS
#include<thread>
#include"Tank.h"

using namespace std;

//Методы класса Tank
Tank::Tank() :Bullet()
{
    hp = 0;
    enemy = false;
    attack = false;
}

Tank::Tank(int xn, int yn, int hpn, int s, int pn, bool en, bool at, BoxCollider bc) :Bullet(xn, yn, s, pn, bc, en)
{
    hp = hpn;
    attack = at;
}

Tank::Tank(const Tank &a) :Bullet(a)
{
    hp = a.hp;
    attack = a.attack;
}

void ShootCooldown(int Cooldown, vector<Bullet> &b, Tank &a)
{
    b.insert(b.end(), Bullet());
    a.SetAttack(true);
    switch (a.GetP())
    {
    case 1:
        b.back().SetX(a.GetX());
        b.back().SetY(a.GetY()+11);
        break;
    case 2:
        b.back().SetX(a.GetX()+11);
        b.back().SetY(a.GetY());
        break;
    case 3:
        b.back().SetX(a.GetX()-11);
        b.back().SetY(a.GetY());
        break;
    case 4:
        b.back().SetX(a.GetX());
        b.back().SetY(a.GetY()-11);
        break;
    }
    b.back().SetP(a.GetP());
    b.back().SetSpeed(5);
    if (a.GetEnemy() == true)
        b.back().SetEnemy(true);
    else
        b.back().SetEnemy(false);
    b.back().Collider.SetBoxCollider(Point(a.GetX() - 2, a.GetY() - 2), Point(a.GetX() + 2, a.GetY() + 2));
    Sleep(Cooldown);
    a.SetAttack(false);
}

void Tank::Shoot(int Cooldown, vector<Bullet> &b, Tank &a)
{
```

```
63      thread Coold(ShootCooldown, Cooldown, ref(b), ref(a));
64      Coold.detach();
65  }
66
67  void Tank::FindTarget(Tank pl, Tank &ii)
68  {
69      thread v(FT, pl, ref(ii));
70      v.detach();
71  }
72
73  void Tank::ChangeCol()
74  {
75      switch (p)
76      {
77      case 1:
78          Collider.SetBoxCollider(Point(x - 10, y - 10), Point(x + 10, y + 11));
79          break;
80      case 2:
81          Collider.SetBoxCollider(Point(x - 10, y - 10), Point(x + 11, y + 10));
82          break;
83      case 3:
84          Collider.SetBoxCollider(Point(x - 11, y - 10), Point(x + 10, y + 10));
85          break;
86      case 4:
87          Collider.SetBoxCollider(Point(x - 10, y - 11), Point(x + 10, y + 10));
88          break;
89      }
90  }
91
92  bool Tank::CheckMoveAi(vector<Wall> &w, vector<Tank> &a, Tank &t)
93  {
94      bool b = false;
95      for (int j = 0; j < w.size(); j++)     //есть ли столкновение со стеной
96      {
97          b = Collider&&w[j].WallCollider;
98          if (b == true) break;
99      }
100     if (b == false) b=Collider&&t.Collider; // столкновение с игроком
101     if (b == false)
102     {
103         for (int i = 0; i < a.size(); i++) //есть ли столкновение с ботов с ботами
104         {
105             b = (Collider&&a[i].Collider) && ((y != a[i].GetY()) || (x != a[i].GetX()));
106             if (b == true) break;
107         }
108     }
109     if (b == false)             //произошёл ли вылет за границы игрового поля
110
111         switch (p)
112         {
113         case 1:
114             b = (y >= 650 - 12);
115             break;
116         case 2:
117             b = (x >= 650 - 12);
118             break;
119         case 3:
120             b = (x <= 50 + 12);
121             break;
122         case 4:
123             b = (y <= 50 + 12);
124             break;
125
126         }
```

```
127     return b;
128 }
129 
130 bool Tank::CheckMove(vector<Wall> &w, vector<Tank> &a)
131 {
132     bool b = false;
133     for (int j = 0; j < w.size(); j++)     //есть ли столкновение со стеной
134     {
135         b = Collider&&w[j].WallCollider;
136         if (b == true) break;
137     }
138     if (b == false)
139     {
140         for (int i = 0; i < a.size(); i++) //есть ли столкновение с ботов с ботами
141         {
142             b = (Collider&&a[i].Collider);
143             if (b == true) break;
144         }
145     }
146     if (b == false)                     //произошёл ли вылет за границы игрового поля
147 
148         switch (p)
149         {
150         case 1:
151             b = (y >= 650 - 12);
152             break;
153         case 2:
154             b = (x >= 650 - 12);
155             break;
156         case 3:
157             b = (x <= 50 + 12);
158             break;
159         case 4:
160             b = (y <= 50 + 12);
161             break;
162 
163         }
164     return b;
165 }
166 
167 void Tank::MoveUp(vector<Wall> &w, vector<Tank> &a)
168 {
169     short int s = 0;
170     p = 1;
171     Collider.SetBoxCollider(Point(x - 10, y - 10), Point(x + 10, y + 11));
172     while (CheckMove(w, a) == false && s != speed)
173     {
174         s++;
175         y = y + 1;
176         ChangeCol();
177     }
178     if (CheckMove(w, a) != false && this->enemy == true)
179         p = 1 + rand() % 4;
180     ChangeCol();
181 }
182 
183 void Tank::MoveRight(vector<Wall> &w, vector<Tank> &a)
184 {
185     short int s = 0;
186     p = 2;
187     Collider.SetBoxCollider(Point(x - 10, y - 10), Point(x + 11, y + 10));
188     while (CheckMove(w, a) == false && s != speed)
189     {
190         s++;
```

```
191         x = x + 1;
192         ChangeCol();
193     }
194     if (CheckMove(w, a) != false && this->enemy == true)
195         p = 1 + rand() % 4;
196     ChangeCol();
197 }
198
199 void Tank::MoveLeft(vector<Wall> &w, vector<Tank> &a)
200 {
201     short int s = 0;
202     p = 3;
203     Collider.SetBoxCollider(Point(x - 11, y - 10), Point(x + 10, y + 10));
204     while (CheckMove(w, a) == false && s != speed)
205     {
206         s++;
207         x = x - 1;
208         ChangeCol();
209     }
210     if (CheckMove(w, a) != false && this->enemy == true)
211         p = 1 + rand() % 4;
212     ChangeCol();
213 }
214
215 void Tank::MoveDown(vector<Wall> &w, vector<Tank> &a)
216 {
217     short int s = 0;
218     p = 4;
219     Collider.SetBoxCollider(Point(x - 10, y - 11), Point(x + 10, y + 10));
220     while (CheckMove(w, a) == false && s != speed)
221     {
222         s++;
223         y = y - 1;
224         ChangeCol();
225     }
226     if (CheckMove(w, a) != false && this->enemy == true)
227         p = 1 + rand() % 4;
228     ChangeCol();
229 }
230
231 void Tank::MoveUpAi(vector<Wall> &w, vector<Tank> &a, Tank &t)
232 {
233     short int s = 0;
234     p = 1;
235     Collider.SetBoxCollider(Point(x - 10, y - 10), Point(x + 10, y + 11));
236     while (CheckMoveAi(w, a,t) == false && s != speed)
237     {
238             s++;
239             y = y + 1;
240             ChangeCol();
241     }
242     if (CheckMoveAi(w, a,t) != false && this->enemy == true)
243         p = 1 + rand() % 4;
244     ChangeCol();
245 }
246
247 void Tank::MoveRightAi(vector<Wall> &w, vector<Tank> &a, Tank &t)
248 {
249     short int s = 0;
250     p = 2;
251     Collider.SetBoxCollider(Point(x - 10, y - 10), Point(x + 11, y + 10));
252     while (CheckMoveAi(w, a, t) == false && s != speed)
253     {
254         s++;
```

```
255         x = x + 1;
256         ChangeCol();
257     }
258     if (CheckMoveAi(w, a, t) != false && this->enemy == true)
259         p = 1 + rand() % 4;
260     ChangeCol();
261 }
262
263 void Tank::MoveLeftAi(vector<Wall> &w, vector<Tank> &a, Tank &t)
264 {
265     short int s = 0;
266     p = 3;
267     Collider.SetBoxCollider(Point(x - 11, y - 10), Point(x + 10, y + 10));
268     while (CheckMoveAi(w, a, t) == false && s != speed)
269     {
270         s++;
271         x = x - 1;
272         ChangeCol();
273     }
274     if (CheckMoveAi(w, a, t) != false && this->enemy == true)
275         p = 1 + rand() % 4;
276     ChangeCol();
277 }
278
279 void Tank::MoveDownAi(vector<Wall> &w, vector<Tank> &a, Tank &t)
280 {
281     short int s = 0;
282     p = 4;
283     Collider.SetBoxCollider(Point(x - 10, y - 11), Point(x + 10, y + 10));
284     while (CheckMoveAi(w, a, t) == false && s != speed)
285     {
286         s++;
287         y = y - 1;
288         ChangeCol();
289     }
290     if (CheckMoveAi(w, a, t) != false && this->enemy == true)
291         p = 1 + rand() % 4;
292     ChangeCol();
293 }
294
295 void Tank::MoveAi(vector<Wall> &w, vector<Tank> &tI, Tank &t)
296 {
297     switch (p)
298     {
299     case 1:
300         this->MoveUpAi(w, tI, t);
301         break;
302     case 2:
303         this->MoveRightAi(w, tI, t);
304         break;
305     case 3:
306         this->MoveLeftAi(w, tI, t);
307         break;
308     case 4:
309         this->MoveDownAi(w, tI, t);
310         break;
311     }
312 }
313
314 void Tank::MovePlayer(Controls k, vector<Wall> &w, vector<Tank> &tI)
315 {
316     if (k.keyPressedUp == true)
317         MoveUp(w, tI);
318     if (k.keyPressedDown == true)
```

```
319         MoveDown(w, tI);
320     if (k.keyPressedRight == true)
321         MoveRight(w, tI);
322     if (k.keyPressedLeft == true)
323         MoveLeft(w, tI);
324 }
325
326 istream &operator >> (istream &in, Tank &c)
327 {
328     in >> c.x >> c.y;
329     in >> c.Collider;
330     in >> c.speed >> c.p >> c.enemy >> c.hp >> c.attack;
331     return in;
332 }
333
334 void FT(Tank pl, Tank &ii)
335 {
336     ii.SetTarget(true);
337     int num;
338     srand(time(NULL));
339     num = rand() % 2;
340     if (num == 0) {
341         if (ii.GetX() > pl.GetX() && ii.GetY() != pl.GetY())ii.SetP(3);
342         if (ii.GetX() < pl.GetX() && ii.GetY() != pl.GetY())ii.SetP(2);
343         if (ii.GetX() == pl.GetX())
344             if (ii.GetY() > pl.GetY()) ii.SetP(4);
345             else ii.SetP(1);
346     }
347     if (num == 1) {
348         if (ii.GetY() > pl.GetY() && ii.GetX() != pl.GetX())ii.SetP(4);
349         if (ii.GetY() < pl.GetY() && ii.GetX() != pl.GetX())ii.SetP(1);
350         if (ii.GetY() == pl.GetY())
351             if (ii.GetX() > pl.GetX()) ii.SetP(3);
352             else ii.SetP(2);
353     }
354     short int j;
355     j = 1000 + rand() % 200;
356     Sleep(j);
357     ii.SetTarget(false);
358 }
```

```h
#pragma once
#include <GL\glut.h>
class Painter
{
public:
    static void DrawTank(int x, int y, int p, bool enemy);
    static void DrawBullet(int x, int y);
    static void DrawWall(int x, int y,int p);
    static void DrawField();
    static void drawText(const char *text, int length, int x, int y);
};
```

```cpp
#include"Painter.h"

void Painter::DrawTank(int x, int y, int p, bool enemy)
{
    int size = 20;
    glBegin(GL_POINTS);
    if (enemy == false)
        glColor3d(0, 0.5, 0);
    else
        glColor3d(0.55, 0.55, 0.55);

    switch (p)
    {
    case 1:
    {
        for (int i = 0; i <= size / 2 - 6; i = i + 2)
        {
            glVertex2f(x - size / 2, y + i);
            glVertex2f(x + size / 2, y + i);
            glVertex2f(x + 1 - size / 2, y + i);
            glVertex2f(x - 1 + size / 2, y + i);
            glVertex2f(x + 2 - size / 2, y + i);
            glVertex2f(x - 2 + size / 2, y + i);
        }
        for (int i = 0; i <= size / 2; i = i + 2)
        {
            glVertex2f(x - size / 2, y - i);
            glVertex2f(x + size / 2, y - i);
            glVertex2f(x + 1 - size / 2, y - i);
            glVertex2f(x - 1 + size / 2, y - i);
            glVertex2f(x + 2 - size / 2, y - i);
            glVertex2f(x - 2 + size / 2, y - i);
        }
        //Крышки гусениц
        for (int i = 0; i <= size / 2; ++i)
        {
            glVertex2f(x + 3 - size / 2, y - i);
            glVertex2f(x - 3 + size / 2, y - i);
        }
        for (int i = 0; i <= size / 2 - 6; ++i)
        {
            glVertex2f(x + 3 - size / 2, y + i);
            glVertex2f(x - 3 + size / 2, y + i);
        }
        //Траки
        for (int i = 2; i <= size - 8; ++i)
        {
            glVertex2f(x + 4 - size / 2, y - size / 2 + i);
            glVertex2f(x - 4 + size / 2, y - size / 2 + i);
            glVertex2f(x + 5 - size / 2, y - size / 2 + i);
            glVertex2f(x - 5 + size / 2, y - size / 2 + i);
        }
        //Корпус танка
        for (int i = 3; i <= size - 7; ++i)
        {
            for (int j = 6; j <= size - 6; j++)
                glVertex2f(x - j + size / 2, y + i - size / 2);
        }
        for (int i = 1; i <= size / 2; ++i)
            glVertex2f(x, y + i);
    }break;
    case 2:
    {
        for (int i = 0; i <= size / 2 - 6; i = i + 2)
```

```
65          {
66              glVertex2f(x + i, y - size / 2);
67              glVertex2f(x + i, y + size / 2);
68              glVertex2f(x + i, y + 1 - size / 2);
69              glVertex2f(x + i, y - 1 + size / 2);
70              glVertex2f(x + i, y + 2 - size / 2);
71              glVertex2f(x + i, y - 2 + size / 2);
72          }
73          for (int i = 0; i <= size / 2; i = i + 2)
74          {
75              glVertex2f(x - i, y - size / 2);
76              glVertex2f(x - i, y + size / 2);
77              glVertex2f(x - i, y + 1 - size / 2);
78              glVertex2f(x - i, y - 1 + size / 2);
79              glVertex2f(x - i, y + 2 - size / 2);
80              glVertex2f(x - i, y - 2 + size / 2);
81          }
82          //Крышки гусениц
83          for (int i = 0; i <= size / 2; ++i)
84          {
85              glVertex2f(x - i, y + 3 - size / 2);
86              glVertex2f(x - i, y - 3 + size / 2);
87          }
88          for (int i = 0; i <= size / 2 - 6; ++i)
89          {
90              glVertex2f(x + i, y + 3 - size / 2);
91              glVertex2f(x + i, y - 3 + size / 2);
92          }
93          //Траки
94          for (int i = 2; i <= size - 8; ++i)
95          {
96              glVertex2f(x - size / 2 + i, y + 4 - size / 2);
97              glVertex2f(x - size / 2 + i, y - 4 + size / 2);
98              glVertex2f(x - size / 2 + i, y + 5 - size / 2);
99              glVertex2f(x - size / 2 + i, y - 5 + size / 2);
100         }
101         //Корпус танка
102         for (int i = 3; i <= size - 7; ++i)
103         {
104             for (int j = 6; j <= size - 6; j++)
105                 glVertex2f(x + i - size / 2, y - j + size / 2);
106         }
107         for (int i = 1; i <= size / 2; ++i)
108             glVertex2f(x + i, y);
109     }break;
110     case 3:
111     {
112         for (int i = 0; i <= size / 2 - 6; i = i + 2)
113         {
114             glVertex2f(x - i, y - size / 2);
115             glVertex2f(x - i, y + size / 2);
116             glVertex2f(x - i, y + 1 - size / 2);
117             glVertex2f(x - i, y - 1 + size / 2);
118             glVertex2f(x - i, y + 2 - size / 2);
119             glVertex2f(x - i, y - 2 + size / 2);
120         }
121         for (int i = 0; i <= size / 2; i = i + 2)
122         {
123             glVertex2f(x + i, y - size / 2);
124             glVertex2f(x + i, y + size / 2);
125             glVertex2f(x + i, y + 1 - size / 2);
126             glVertex2f(x + i, y - 1 + size / 2);
127             glVertex2f(x + i, y + 2 - size / 2);
128             glVertex2f(x + i, y - 2 + size / 2);
```

```
129         }
130         //Крышки гусениц
131         for (int i = 0; i <= size / 2; ++i)
132         {
133             glVertex2f(x + i, y + 3 - size / 2);
134             glVertex2f(x + i, y - 3 + size / 2);
135         }
136         for (int i = 0; i <= size / 2 - 6; ++i)
137         {
138             glVertex2f(x - i, y + 3 - size / 2);
139             glVertex2f(x - i, y - 3 + size / 2);
140         }
141         //Траки
142         for (int i = 2; i <= size - 8; ++i)
143         {
144             glVertex2f(x + size / 2 - i, y + 4 - size / 2);
145             glVertex2f(x + size / 2 - i, y - 4 + size / 2);
146             glVertex2f(x + size / 2 - i, y + 5 - size / 2);
147             glVertex2f(x + size / 2 - i, y - 5 + size / 2);
148         }
149         //Корпус танка
150         for (int i = 3; i <= size - 7; ++i)
151         {
152             for (int j = 6; j <= size - 6; j++)
153                 glVertex2f(x - i + size / 2, y + j - size / 2);
154         }
155         for (int i = 1; i <= size / 2; ++i)
156             glVertex2f(x - i, y);
157     }break;
158     case 4:
159     {
160         for (int i = 0; i <= size / 2 - 6; i = i + 2)
161         {
162             glVertex2f(x - size / 2, y - i);
163             glVertex2f(x + size / 2, y - i);
164             glVertex2f(x + 1 - size / 2, y - i);
165             glVertex2f(x - 1 + size / 2, y - i);
166             glVertex2f(x + 2 - size / 2, y - i);
167             glVertex2f(x - 2 + size / 2, y - i);
168         }
169         for (int i = 0; i <= size / 2; i = i + 2)
170         {
171             glVertex2f(x - size / 2, y + i);
172             glVertex2f(x + size / 2, y + i);
173             glVertex2f(x + 1 - size / 2, y + i);
174             glVertex2f(x - 1 + size / 2, y + i);
175             glVertex2f(x + 2 - size / 2, y + i);
176             glVertex2f(x - 2 + size / 2, y + i);
177         }
178         //Крышки гусениц
179         for (int i = 0; i <= size / 2; ++i)
180         {
181             glVertex2f(x + 3 - size / 2, y + i);
182             glVertex2f(x - 3 + size / 2, y + i);
183         }
184         for (int i = 0; i <= size / 2 - 6; ++i)
185         {
186             glVertex2f(x + 3 - size / 2, y - i);
187             glVertex2f(x - 3 + size / 2, y - i);
188         }
189         //Траки
190         for (int i = 2; i <= size - 8; ++i)
191         {
192             glVertex2f(x + 4 - size / 2, y + size / 2 - i);
```

```
193             glVertex2f(x - 4 + size / 2, y + size / 2 - i);
194             glVertex2f(x + 5 - size / 2, y + size / 2 - i);
195             glVertex2f(x - 5 + size / 2, y + size / 2 - i);
196         }
197         //Корпус танка
198         for (int i = 3; i <= size - 7; ++i)
199         {
200             for (int j = 6; j <= size - 6; j++)
201                 glVertex2f(x + j - size / 2, y - i + size / 2);
202         }
203         for (int i = 1; i <= size / 2; ++i)
204             glVertex2f(x, y - i);
205     }break;
206     }
207     glEnd();
208 }
209
210 void Painter::DrawBullet(int x, int y)
211 {
212     glBegin(GL_POINTS);
213     glColor3d(0, 0, 0);
214     glVertex2f(x, y);
215     glVertex2f(x + 1, y);
216     glVertex2f(x - 1, y);
217     glVertex2f(x, y + 1);
218     glVertex2f(x, y - 1);
219     glEnd();
220 }
221
222 void Painter::DrawWall(int x, int y, int p)
223 {
224
225     switch (p)
226     {
227         //Непробиваемая стена
228     case 1:
229     {
230         glBegin(GL_QUADS);
231         glColor3d(1, 1, 1);
232         glVertex2f(x - 15, y + 15);
233         glVertex2f(x - 15, y - 15);
234         glVertex2f(x + 15, y - 15);
235         glVertex2f(x + 15, y + 15);
236         glEnd();
237         break;
238     }
239         //Кирпичная пробиваемая стена
240     case 2:
241     {
242         glBegin(GL_QUADS);// 0.95, 0.91, 0.788
243         glColor3d(1, 0.15, 0.05);
244         glVertex2f(x - 15, y + 15);
245         glVertex2f(x - 15, y - 15);
246         glVertex2f(x + 15, y - 15);
247         glVertex2f(x + 15, y + 15);
248         glEnd();
249
250         glBegin(GL_QUADS);
251         glColor3d(0.95, 0.95, 0.8);
252         glVertex2f(x - 9, y + 15);
253         glVertex2f(x - 9, y - 15);
254         glVertex2f(x - 11, y - 15);
255         glVertex2f(x - 11, y + 15);
256         glEnd();
```

```
257
258         glBegin(GL_QUADS);
259         glColor3d(0.95, 0.95, 0.8);
260         glVertex2f(x - 9 + 10, y + 15);
261         glVertex2f(x - 9 + 10, y - 15);
262         glVertex2f(x - 11 + 10, y - 15);
263         glVertex2f(x - 11 + 10, y + 15);
264         glEnd();
265
266         glBegin(GL_QUADS);
267         glColor3d(0.95, 0.95, 0.8);
268         glVertex2f(x - 9 + 20, y + 15);
269         glVertex2f(x - 9 + 20, y - 15);
270         glVertex2f(x - 11 + 20, y - 15);
271         glVertex2f(x - 11 + 20, y + 15);
272         glEnd();
273
274         glBegin(GL_QUADS);
275         glColor3d(0.95, 0.95, 0.8);
276         glVertex2f(x - 15, y - 11 + 10);
277         glVertex2f(x - 15, y - 9 + 10);
278         glVertex2f(x + 15, y - 9 + 10);
279         glVertex2f(x + 15, y - 11 + 10);
280         glEnd();
281
282         glBegin(GL_QUADS);
283         glColor3d(0.95, 0.95, 0.8);
284         glVertex2f(x - 15, y - 11 + 20);
285         glVertex2f(x - 15, y - 9 + 20);
286         glVertex2f(x + 15, y - 9 + 20);
287         glVertex2f(x + 15, y - 11 + 20);
288         glEnd();
289
290         glBegin(GL_QUADS);
291         glColor3d(0.95, 0.95, 0.8);
292         glVertex2f(x - 15, y - 11);
293         glVertex2f(x - 15, y - 9);
294         glVertex2f(x + 15, y - 9);
295         glVertex2f(x + 15, y - 11);
296         glEnd();
297         break;
298     }
299         //Вода
300     case 3:
301     {
302         glBegin(GL_QUADS);
303         glColor3d(0, 0.5, 1);
304         glVertex2f(x - 15, y + 15);
305         glVertex2f(x - 15, y - 15);
306         glVertex2f(x + 15, y - 15);
307         glVertex2f(x + 15, y + 15);
308         glEnd();
309
310         glBegin(GL_QUADS);
311         glColor3d(0.5, 0.775, 1);
312         glVertex2f(x - 12, y -10);
313         glVertex2f(x - 12, y - 12);
314         glVertex2f(x +1, y - 12);
315         glVertex2f(x - 5, y -10);
316         glEnd();
317
318         glBegin(GL_QUADS);
319         glColor3d(0.5, 0.775, 1);
320         glVertex2f(x - 12+15, y - 10+4);
```

```
321         glVertex2f(x - 12+15, y - 12+4);
322         glVertex2f(x + 2+15, y - 12+4);
323         glVertex2f(x - 5+15, y - 10+4);
324         glEnd();
325 
326         glBegin(GL_QUADS);
327         glColor3d(0.5, 0.775, 1);
328         glVertex2f(x - 12 + 10, y - 10 + 10);
329         glVertex2f(x - 12 + 10, y - 12 + 10);
330         glVertex2f(x -5 + 10, y - 12 + 10);
331         glVertex2f(x +2 + 10, y - 10 + 10);
332         glEnd();
333 
334         glBegin(GL_QUADS);
335         glColor3d(0.5, 0.775, 1);
336         glVertex2f(x - 14 + 2, y - 10 + 6);
337         glVertex2f(x - 14 + 2, y - 12 + 6);
338         glVertex2f(x - 7 + 2, y - 12 + 6);
339         glVertex2f(x + 2, y - 10 + 6);
340         glEnd();
341 
342         glBegin(GL_QUADS);
343         glColor3d(0.5, 0.775, 1);
344         glVertex2f(x - 6 + 8, y - 10 + 20);
345         glVertex2f(x - 6 + 8, y - 12 + 20);
346         glVertex2f(x + 6 + 8, y - 12 + 20);
347         glVertex2f(x + 2 + 8, y - 10 + 20);
348         glEnd();
349 
350         glBegin(GL_QUADS);
351         glColor3d(0.5, 0.775, 1);
352         glVertex2f(x - 14 + 5, y - 10 + 15);
353         glVertex2f(x - 14 + 8, y - 12 + 15);
354         glVertex2f(x + 8, y - 12 + 15);
355         glVertex2f(x + 4+8, y - 10 + 15);
356         glEnd();
357 
358         glBegin(GL_QUADS);
359         glColor3d(0.5, 0.775, 1);
360         glVertex2f(x - 14, y - 10 + 22);
361         glVertex2f(x - 13, y - 12 + 22);
362         glVertex2f(x + 1, y - 12 + 22);
363         glVertex2f(x -1, y - 10 + 21);
364         glEnd();
365     }
366     }
367 }
368 
369 void Painter::DrawField()
370 {
371     glBegin(GL_LINES);
372     glColor3d(1, 1, 1);
373     glVertex3d(50, 50, 0);
374     glVertex3d(50, 650, 0);
375     glVertex3d(650, 50, 0);
376     glVertex3d(650, 650, 0);
377     glVertex3d(50, 50, 0);
378     glVertex3d(650, 50, 0);
379     glVertex3d(50, 650, 0);
380     glVertex3d(650, 650, 0);
381     glEnd();
382     glBegin(GL_QUADS);
383     glColor3d(0.85, 0.85, 0);
384     glVertex2f(51, 649);
```

```
385     glVertex2f(51, 51);
386     glVertex2f(649, 51);
387     glVertex2f(649, 649);
388     glEnd();
389 }
390 
391 void Painter::drawText(const char *text, int length, int x, int y)
392 {
393     glColor3d(1, 1, 1);
394     glMatrixMode(GL_PROJECTION);
395     double *matrix = new double[16];
396     glGetDoublev(GL_PROJECTION_MATRIX, matrix);
397     glLoadIdentity();
398     glOrtho(0, 800, 0, 700, -5, 5);
399     glMatrixMode(GL_MODELVIEW);
400     glLoadIdentity();
401     glPushMatrix();
402     glLoadIdentity();
403     glRasterPos2i(x, y);
404     for (int i = 0; i<length; i++)
405     {
406         glutBitmapCharacter(GLUT_BITMAP_HELVETICA_18, (int)text[i]);
407     }
408     glPopMatrix();
409     glMatrixMode(GL_PROJECTION);
410     glLoadMatrixd(matrix);
411     glMatrixMode(GL_MODELVIEW);
412     delete[] matrix;
413 }
```

```
#pragma once

class Controls
{
public:
    bool keyPressedUp;
    bool keyPressedDown;
    bool keyPressedRight;
    bool keyPressedLeft;
    bool keyPressedFire;

    Controls();

    static void KeyboardDown(unsigned char k, int x, int y);
    static void KeyboardUp(unsigned char k, int x, int y);
    static void KeyboardSpecialUp(int k, int x, int y);
    static void KeyboardSpecialDown(int key, int, int);

    static void SetKeyPressedUp(Controls &k, bool s) { k.keyPressedUp = s; }
    static void SetKeyPressedDown(Controls &k, bool s) { k.keyPressedDown = s; }
    static void SetKeyPressedRight(Controls &k, bool s) { k.keyPressedRight = s; }
    static void SetKeyPressedLeft(Controls &k, bool s) { k.keyPressedLeft = s; }
    static void SetKeyPressedFire(Controls &k, bool s) { k.keyPressedFire = s; }
};
```

```cpp
#include"Control.h"
#include<windows.h>
#include"Game.h"

extern Game G1;

Controls::Controls()
{
    keyPressedUp = 0;
    keyPressedDown = 0;
    keyPressedRight = 0;
    keyPressedLeft = 0;
    keyPressedFire = 0;
}

void Controls::KeyboardSpecialDown(int key, int, int)
{
    switch (key)
    {
    case GLUT_KEY_LEFT:
        SetKeyPressedUp(G1.keyboard, false);
        SetKeyPressedDown(G1.keyboard, false);
        SetKeyPressedRight(G1.keyboard, false);
        SetKeyPressedLeft(G1.keyboard, true);
        break;
    case GLUT_KEY_UP:
        SetKeyPressedUp(G1.keyboard, true);
        SetKeyPressedDown(G1.keyboard, false);
        SetKeyPressedRight(G1.keyboard, false);
        SetKeyPressedLeft(G1.keyboard, false);
        break;
    case GLUT_KEY_RIGHT:
        SetKeyPressedUp(G1.keyboard, false);
        SetKeyPressedDown(G1.keyboard, false);
        SetKeyPressedRight(G1.keyboard, true);
        SetKeyPressedLeft(G1.keyboard, false);
        break;
    case GLUT_KEY_DOWN:
        SetKeyPressedUp(G1.keyboard, false);
        SetKeyPressedDown(G1.keyboard, true);
        SetKeyPressedRight(G1.keyboard, false);
        SetKeyPressedLeft(G1.keyboard, false);
        break;
    }
}

void Controls::KeyboardDown(unsigned char k, int x, int y)
{
    switch (k)
    {
    case 'w':
        SetKeyPressedUp(G1.keyboard, true);
        SetKeyPressedDown(G1.keyboard, false);
        SetKeyPressedRight(G1.keyboard, false);
        SetKeyPressedLeft(G1.keyboard, false);
        break;
    case 'd':
        SetKeyPressedUp(G1.keyboard, false);
        SetKeyPressedDown(G1.keyboard, false);
        SetKeyPressedRight(G1.keyboard, true);
        SetKeyPressedLeft(G1.keyboard, false);
        break;
    case 'a':
        SetKeyPressedUp(G1.keyboard, false);
```

```
65          SetKeyPressedDown(G1.keyboard, false);
66          SetKeyPressedRight(G1.keyboard, false);
67          SetKeyPressedLeft(G1.keyboard, true);
68          break;
69      case 's':
70          SetKeyPressedUp(G1.keyboard, false);
71          SetKeyPressedDown(G1.keyboard, true);
72          SetKeyPressedRight(G1.keyboard, false);
73          SetKeyPressedLeft(G1.keyboard, false);
74          break;
75      case 'l':
76          SetKeyPressedFire(G1.keyboard, true);
77          break;
78      case 'r':
79          G1.Restart();
80          break;
81      case 27:
82          exit(0);
83          break;
84      }
85  }
86
87  void Controls::KeyboardSpecialUp(int k, int x, int y)
88  {
89      switch (k)
90      {
91      case GLUT_KEY_LEFT:
92          SetKeyPressedLeft(G1.keyboard, false);
93      break;
94      case GLUT_KEY_UP:
95          SetKeyPressedUp(G1.keyboard, false);
96          break;
97      case GLUT_KEY_RIGHT:
98          SetKeyPressedRight(G1.keyboard, false);
99          break;
100     case GLUT_KEY_DOWN:
101         SetKeyPressedDown(G1.keyboard, false);
102         break;
103     }
104 }
105
106 void Controls::KeyboardUp(unsigned char k, int x, int y)
107 {
108     switch (k)
109     {
110     case 'w':
111         SetKeyPressedUp(G1.keyboard, false);
112         break;
113
114     case 'd':
115         SetKeyPressedRight(G1.keyboard, false);
116         break;
117
118     case 'a':
119         SetKeyPressedLeft(G1.keyboard, false);
120         break;
121
122     case 's':
123         SetKeyPressedDown(G1.keyboard, false);
124         break;
125
126     case 'l':
127         SetKeyPressedFire(G1.keyboard, false);
128         break;
```

```
129
130     case 27:
131         exit(0);
132         break;
133     }
134 }
```

```h
#pragma once
#include "Painter.h"
#include "Tank.h"

using namespace std;

class Game
{
public:
    const int ScreenSizeX = GetSystemMetrics(SM_CXSCREEN);
    const int ScreenSizeY = GetSystemMetrics(SM_CYSCREEN);
    const int WindowSizeX = 800;    //Размер окна с игрой
    const int WindowSizeY = 700;    //Размер окна с игрой

    short int Score;    //Счёт
    short int Waves;    //Волны
    short int state;    //Состояние игры 0-обычная 1-Поражение 2-Новая волна
    //Переменные необходимые для подсчёта FPS
    char FPS[10];
    int frame;
    int time;
    int timebase;

    Tank t;                         //танк игрока
    vector<Bullet> ArrayOfBullet;   //массив всех пуль в игре
    vector<Tank> ArrayOfBots;       //массив всех ботов
    vector<Wall> ArrayOfWall;       //массив всех стен
    Controls keyboard;              //подключение управления

    Game();
    void Restart();             //перезапуск игры после поражения
    void Respawn();             //респавн игрока после попадания в него пули
    void GameOver();            //Поражение
    void LoadLevel();           //Загрузка уровня
    void OtherGameObjects();    //Обработка всего, что происходит с другими объектами
    void Bullets();             //Обработка всего, что происходит с пулями
    void Player();              //Обработка всего, что происходит с игроком
    void Bots();                //Обработка всего, что происходит с ботами
};
```

```cpp
#include"Game.h"
#include<thread>

void NewWave(vector<Tank> &ArrayOfBots, short int & state, short int &Score, short int &Waves, short int h)
{
    Sleep(3000);
    Tank b;
    fstream f2("2.txt", ios::in);
    while (!f2.eof())
    {
        f2 >> b;
        ArrayOfBots.push_back(b);
    };
    f2.close();
    state = 0;
    Score = Score + Waves * 100 + 20 * h;
    Waves++;
}

char* IntToStr(int n)
{
    char temp[10];
    sprintf(temp, "%d", n);
    return(temp);
}

Game::Game()
{
    t = Tank(351, 206, 5, 1, 1, false, false, BoxCollider(Point(351-10, 206-10), Point(351+10, 206+11)));
    Score = 0;
    frame = 0;
    time = 0;
    timebase = 0;
    Waves = 1;
    state = 0;
    LoadLevel();
}

void Game::GameOver()
{
    Painter::drawText("Game Over", 9, 345, 450);
    char t1[20] = "Score: ";
    strcat(t1, IntToStr(Score));
    Painter::drawText(t1, 20, 355, 400);
    char t2[20] = "Waves: ";
    strcat(t2, IntToStr(Waves));
    Painter::drawText(t2, 20, 355, 350);
    Painter::drawText("Press r to restart or ESC to exit", 33, 270, 300);
}

void Game::OtherGameObjects()
{
    Painter::DrawField();   //Отрисовываем игровое поле

    char s1[20] = "Score: ";                //Отрисовываем элементы интерфейса
    strcat(s1, IntToStr(Score));
    Painter::drawText(s1, 20, 660, 600-20);

    char s2[20] = "Health: ";
    strcat(s2, IntToStr(t.GetHp()));
    Painter::drawText(s2, 20, 660, 550-20);

```

```
63      char s3[20] = "Speed: ";
64      strcat(s3, IntToStr(t.GetSpeed()));
65      Painter::drawText(s3, 20, 660, 500-20);
66
67      char s4[20] = "Waves: ";
68      strcat(s4, IntToStr(Waves));
69      Painter::drawText(s4, 20, 660, 450-20);
70
71      char s5[20] = "Enemies: ";
72      strcat(s5, IntToStr(ArrayOfBots.size()));
73      Painter::drawText(s5, 20, 660, 400-20);
74                                                  //Подсчёт FPS
75      frame++;
76      time = glutGet(GLUT_ELAPSED_TIME);
77      if (time - timebase > 1000) {
78          sprintf(FPS, "FPS: %4.2f",
79              frame*1000.0 / (time - timebase));
80          timebase = time;
81          frame = 0;
82      }
83      Painter::drawText(FPS, 10, 660, 650-20);
84                                                  //Отрисовка стен
85      for (int q = 0; q < ArrayOfWall.size(); q++)
86          Painter::DrawWall(ArrayOfWall[q].GetX(), ArrayOfWall[q].GetY(), ArrayOfWall[q].GetType());
              //Отрисовываем стены
87  }
88
89  void Game::Bots()
90  {
91      for (int i = 0; i < ArrayOfBots.size(); i++)
92      {
93          Painter::DrawTank(ArrayOfBots[i].GetX(), ArrayOfBots[i].GetY(), ArrayOfBots[i].GetP(),
              ArrayOfBots[i].GetEnemy());  //Отрисовываем бота
94
95          if (ArrayOfBots[i].GetTarget() == false)ArrayOfBots[i].FindTarget(t, ArrayOfBots[i]);
96          ArrayOfBots[i].MoveAi(ArrayOfWall, ArrayOfBots, t);                                 //Движение
              бота
97          short int cd = 1500 + rand() % 2000;
98          if (ArrayOfBots[i].GetAttack() == false)
99              ArrayOfBots[i].Shoot(cd, ArrayOfBullet, ArrayOfBots[i]);                 //Стрельба бота
100     }
101 }
102
103 void Game::Player()
104 {
105     if (t.GetHp() > 0)
106         Painter::DrawTank(t.GetX(), t.GetY(), t.GetP(), t.GetEnemy());  //Отрисовываем игрока
107     else
108     {
109         ArrayOfBots.clear();
110         ArrayOfBullet.clear();
111         ArrayOfWall.clear();
112         glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
113         state = 1;
114     }
115     t.MovePlayer(keyboard, ArrayOfWall, ArrayOfBots);                   //Движение игрока
116
117     if (t.GetAttack() == false && keyboard.keyPressedFire == true)
118         t.Shoot(1000, ArrayOfBullet, t);                                //Стрельба игрока
119 }
120
121 void Game::Bullets()
122 {
123     for (int i = 0; i < ArrayOfBullet.size(); i++)
```

```
124     {
125         Painter::DrawBullet(ArrayOfBullet[i].GetX(), ArrayOfBullet[i].GetY());  //Отрисовываем пули
126         ArrayOfBullet[i].MoveBullet();                                           //Движение пуль
127 
128         for (int l = 0; l < ArrayOfWall.size(); l++)                             //Стрельба по стене
129         {
130             if (ArrayOfBullet[i].Collider&&ArrayOfWall[l].WallCollider && (ArrayOfWall[l].GetType()
                == 1 || ArrayOfWall[l].GetType() == 2))
131             {
132                 ArrayOfBullet.erase(ArrayOfBullet.begin() + i);                  //Удаляем пулю при
                    столкновении со стеной
133                 if (ArrayOfWall[l].GetType() == 2)
134                     ArrayOfWall.erase(ArrayOfWall.begin() + l);
135             }
136         }
137 
138         for (int s = 0; s < ArrayOfBots.size(); s++)                    //Стрельба по ботам
139         {
140             if ((ArrayOfBullet[i].Collider&&ArrayOfBots[s].Collider) && ArrayOfBullet[i].GetEnemy()
                == false)
141             {
142                 ArrayOfBullet.erase(ArrayOfBullet.begin() + i);
143                 ArrayOfBots[s].SetHp(ArrayOfBots[s].GetHp() - 1);
144                 if (ArrayOfBots[s].GetHp() == 0)
145                 {
146                     Score = Score + 100* ArrayOfBots[s].GetSpeed();
147                     ArrayOfBots.erase(ArrayOfBots.begin() + s);
148                     if (ArrayOfBots.size() == 0 && state != 2)
149                     {
150                         state = 2;
151                         thread Coold1(NewWave, ref(ArrayOfBots), ref(state), ref(Score), ref(Waves),
                            t.GetHp());
152                         Coold1.detach();
153                     }
154                 }
155             }
156         }
157 
158         if (ArrayOfBullet[i].Collider&&t.Collider&&ArrayOfBullet[i].GetEnemy() == true) //Стрельба по
                игроку
159         {
160             ArrayOfBullet.erase(ArrayOfBullet.begin() + i);
161             t.SetHp(t.GetHp() - 1);
162             Respawn();
163         }
164 
165         //Вылет пули за пределы экрана
166         if (ArrayOfBullet[i].GetX() >= 650 || ArrayOfBullet[i].GetX() <= 50 || ArrayOfBullet[i].GetY
            () >= 650 || ArrayOfBullet[i].GetY() <= 50)
167             ArrayOfBullet.erase(ArrayOfBullet.begin() + i);
168     }
169 }
170 
171 void Game::LoadLevel()
172 {
173     char ch = 0;
174     int i = 0;
175     Wall w;
176     fstream f1("1.txt", ios::in);
177     while (!f1.eof())
178     {
179         f1 >> w;
180         ArrayOfWall.push_back(w);
181     };
```

```
182     f1.close();
183     Tank b;
184     fstream f2("2.txt", ios::in);
185     while (!f2.eof())
186     {
187         f2 >> b;
188         ArrayOfBots.push_back(b);
189     };
190     f2.close();
191 }
192
193 void Game::Restart()
194 {
195     if (state == 1)
196     {
197         state = 0;
198         t = Tank(351, 206, 5, 1, 1, false, false, BoxCollider(Point(351 - 10, 206 - 10), Point(351 +
              10, 206 + 11)));
199         LoadLevel();
200     }
201 }
202
203 void Game::Respawn()
204 {
205     t.SetX(351);
206     t.SetY(206);
207     t.SetP(1);
208     t.Collider.SetBoxCollider(Point(351 - 10, 206 - 10), Point(351 + 10, 206 + 11));
209 }
```

```cpp
#include"Game.h"
#include <GL\glut.h>

Game G1;

void renderScene(void)
{
    glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
    if (G1.state == 0)
    {
        G1.OtherGameObjects();
        G1.Player();
        G1.Bots();
        G1.Bullets();
    }
    if (G1.state == 1)
        G1.GameOver();
    if (G1.state == 2)
    {
        G1.OtherGameObjects();
        G1.Player();
        G1.Bullets();
    }
    glutSwapBuffers();
}

void Timer(int)
{
    renderScene();
    glutTimerFunc(15, Timer, 1);
}

int main(int argc, char **argv)
{
    glutInit(&argc, argv);  // Инициализация GLUT
    glutInitDisplayMode(GLUT_DOUBLE | GLUT_RGB);//устанавливаем палитру цветов RGB и двойной буфер
    glutInitWindowPosition(G1.ScreenSizeX/2- G1.WindowSizeX/2, G1.ScreenSizeY/2- G1.WindowSizeY/2); // ↵
      устанавливаем позицию окна
    glutInitWindowSize(G1.WindowSizeX, G1.WindowSizeY);//Устанавливает размер окна
    glClearColor(0, 0, 0, 1.0);                    //задаём значение очистки цветом буфера цвета
    glutCreateWindow("Танчики");                   //создание окна "Танчики"
    glMatrixMode(GL_PROJECTION);                   //говорит о том, что команды относятся к проекту.
    glLoadIdentity();                              //считывает текущую матрицу
    glOrtho(0, 800, 0, 700, -1, 1);                //установка ортогональной проекции
    glutDisplayFunc(renderScene);                  //регистрация обратных вызовов
    glutKeyboardFunc(&Controls::KeyboardDown);     //регистрация нажатий кнопок на клаве
    glutKeyboardUpFunc(&Controls::KeyboardUp);     //регистрация отжатий кнопок на клаве
    glutSpecialFunc(&Controls::KeyboardSpecialDown);//регистрация нажатий спец кнопок
    glutSpecialUpFunc(&Controls::KeyboardSpecialUp);//регистрация отжатий спец кнопок
    Timer(0);                                      //устанавливаем таймер
    glutMainLoop();     // Основной цикл GLUT
    return 0;
}
```