#### Point MenuBase enum Direction Поля Поля std::string title; std::vector<std::string> items; LEFT, UP, RIGHT, DOWN float x, y; int position; Методы Controller\* c; Struct Color Point(); Методы Point(float, float); float GetX() const; MenuBase(std::string title, Controller\* c); float GetY() const; Поля void Translate(float x, float y); void Keyboard(int key); void SpecialKeyboard(int key); void Render(); Color(float \_r, float \_g, float \_b) {r = \_r; g = virtual void SelectMenuItem(int) = 0; \_g; b = \_b;} float g; float b; GameObject Rectangle LoseMenu Menu PauseMenu Menu Bot Поля Поля Поля Поля Поля Поля Поля float width, height; std::string originalTitle; ----bool dead; ----------Tank tank; std::vector<int> keys; Методы Методы Методы Методы Методы Методы int shootKey; std::list<int> operationQueue; int opDelay, perfDelay; Pause Menu(std::string title, Game Object(); Rectangle(); Lose Menu(std::string title, Controller\* c); Menu(std::string title, Controller\* c); Menu(std::string title, Controller\* c); std::chrono::system\_clock::time\_point GameObject(float \_x, float \_y); Controller\*c); Rectangle(float, float, float, float); void SetScore(int score); void SelectMenuItem(int); void SelectMenuItem(int); prevOp; void SelectMenuItem(int); virtual void Tick() = 0; Rectangle (const Rectangle & other); void SelectMenultem(int); std::chrono::system\_clock::time\_point virtual void Render() = 0; float GetWidth(); virtual void Keyboard(int key) = 0; float GetHeight(); virtual void SpecialKeyboard(int key) = 0; bool Intersects(const Rectangle & other) Методы bool Is Dead(); Rectangle GetRect() const; Bot(Tank t); void Tick(); void Render(); void Emulate Keyboard(); Tank& GetTank(); void SetOpDelay(int delay); Controller Game Object && void SetPerfDelay(int delay); int GetOpDelay(); int GetPerfDelay(); Поля int score;

# **Мар** Поля

std::list<Block> blocks;

## Методы

Map(const std::string& filename, float, float); bool Intersects(const Rectangle& r); std::list<Block>& GetBlocks(); void Tick(); void Render(); void Keyboard(int); void Keyboard(float, float); void SpecialKeyboard(int); void SpecialKeyboard(int); void ReadMapFromFile(const std::string& filename);

Поля

Color color;
bool destructible;
int hp;

Mетоды

Block(float, float, float, float, Color, bool, int \_hp = 0);
void Tick();
void Render();
void Keyboard(int);
void SpecialKeyboard(int);
bool Is Destructible() const;

int GetHp() const;

void SetHp(int);

Block

# Bullet Поля float speed; Direction dir;

float speed; Direction dir;

Bullet();
Bullet(float x, float y, Direction dir);
void Tick();
void Render();
void Keyboard(int);
void SpecialKeyboard(int);

# Player Tank

Поля

Методы

void Tick();

void Render();

# Поля int hp; const double STEP\_TIME = 0.1;

const double RELOAD\_TIME = 0.3;
float restoreX, restoreY;
bool isEnemy;
Direction prevDir;
Direction dir;
Controller\* c;
std::vector<Map> collisionMaps;
std::chrono::high\_resolution\_clock::time\_poi
nt prevStepTime;
std::chrono::high\_resolution\_clock::time\_poi
nt prevShootTime;

#### Методы

Tank();
Tank(float \_x, float \_y, std::string texfolder,
Controller& \_c, bool enemy, int hp);
Tank(const Tank& other);
Tank& operator=(const Tank&);
Map& GetCollisionMap();
void Render();
void SpecialKeyboard(int key);
void Keyboard(int key);
void Restore();
void Shoot();
void Tick();
void Tick();
void Hit();
bool Is Enemy();
int GetHp();
void Move(Direction d, float dx, float dy);
void Copy(const Tank& t);

std::vector<Bullet> playerBullets; std::vector<Bullet> enemyBullets; std::vector<Bot> enemies; Menu\* menu; PauseMenu\* pmenu; LoseMenu\* lmenu; Tank\* player; Map\* map; bool pause; bool start;

bool lose;

void Tick();

void Render();

### Методы

void NewGame();
void Continue();
void StopGame();
void ResetGame();
void AddPlayerBullet(Bullet b);
void AddEnemyBullet(Bullet b);
void SetPlayer(Tank\* p);
void SetMap(Map\* m);
void SetMenu(Menu\* m);
void SetPlayerMenu\* m);
void SetPlayerMenu\* m);

static Controller & getInstance();

void SetLoseMenu(LoseMenu\* m); void AddBot(Bot b); void Keyboard(unsigned char key); void SpecialKeyboard(int k);

~Controller(); Controller(Controller const&) = delete; Controller& operator= (Controller const&) = delete; void HandleBlocks();

void HandleBlocks(); void HandleEnemies(); void HandleBullets();

void RenderScore(); Controller();

void HandleBullets(); bool CollidesBlocks(const Rectangle& r);

# Примечания:

Класс

Абстрактный Класс

Перечисление(enum)

Структура

Наследование

Виртуальное Наследование