Министерство образования и науки Российской Федерации   
Федеральное государственное бюджетное образовательное учреждение высшего образования   
«Петрозаводский государственный университет»  
 Физико-технический институт   
Кафедра информационно-измерительных систем и физической электроники

КУРСОВОЙ ПРОЕКТ   
по дисциплине «Технология программирования»

Авторы работы:   
студенты группы 21316   
\_\_\_\_\_\_\_\_\_\_\_ И. В. Сосновский  
\_\_\_\_\_\_\_\_\_\_\_ С.Н. Турцевич  
\_\_\_\_\_\_\_\_\_\_\_ Е.В. Матвеев  
\_\_\_\_\_\_\_\_\_\_\_Э.А. Шаблинский  
«\_\_\_ »\_\_\_\_\_\_\_\_ 2017 г.

Научный руководитель:   
канд. физ.-мат. наук, доцент  
 \_\_\_\_\_\_\_\_\_\_\_ А.В. Бульба  
 «\_\_\_ » \_\_\_\_\_\_\_\_2017 г.

Петрозаводск 2017

Цель: придумать сюжет игры и реализовать его при помощи Microsoft Visual Studio C++, использую объектно-ориентированное программирование и SFML, и системы контроля версий Git, а также использовать при реализации диаграммы классов UML.

ОПИСАНИЕ СЮЖЕТА ИГРЫ:

Главный герой, попадает в мир орков и должен в нём выжить. Он получает урон от атак орков и от стрел башен, располагаемых случайно на карте, в зависимости от движения игрока. Герой может блокировать атаки врагов и не получать урон, но при этом не может атаковать и двигаться. Также игрок может «сгореть» в лаве, если будет долго стоять на ней (лава отнимает здоровье). Также случайно на карте появляются 2 вида бонусов: повышение атаки героя и повышение его здоровья, которые должны быть собраны героем, чтобы получить их эффект. При атаке героем орков и башен ему начисляются очки. Суть игры: набрать как можно больше очков.

ПРОГРАММНАЯ РЕАЛИЗАЦИЯ:

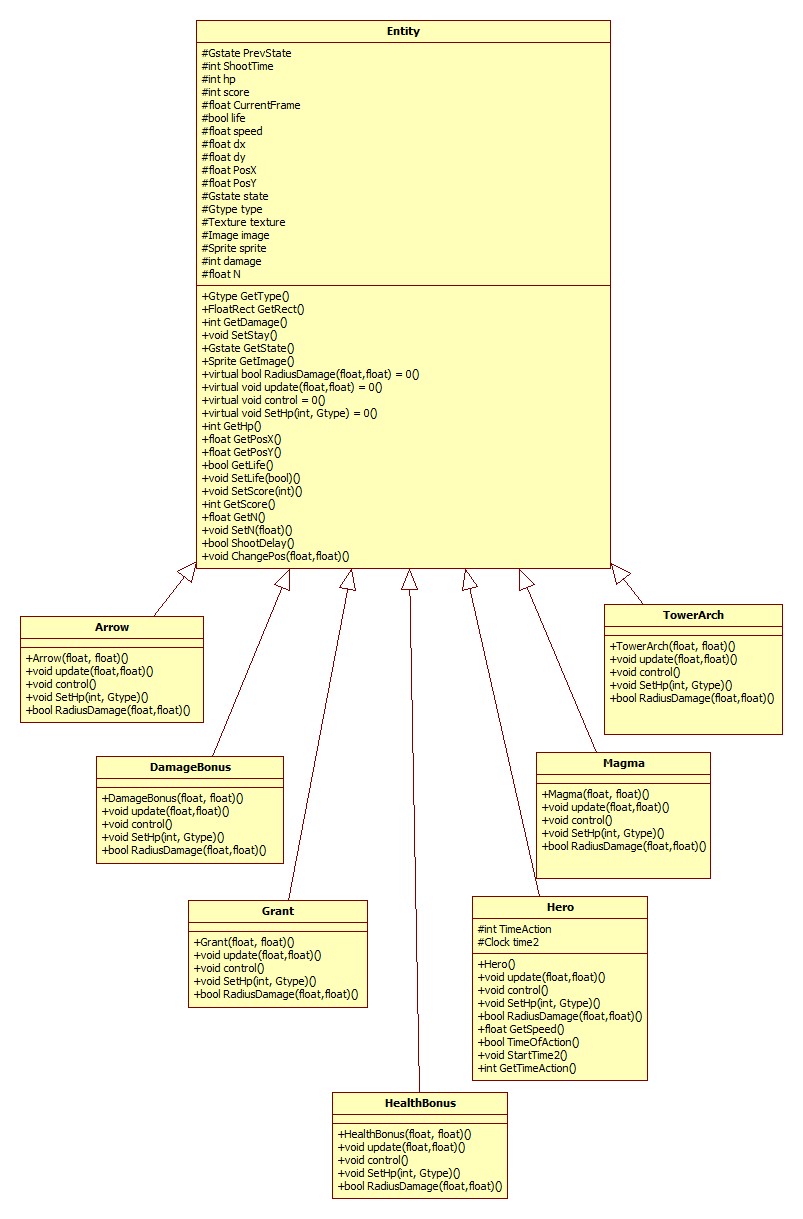
Среда разработки: Microsoft Visual Studio

Язык программирования: C++

Заголовочные файлы:  
arrow.h - интерфейс класса Arrow;  
DamageBonus.h - интерфейс класса DamageBonus;  
Entity.h –интерфейс класса Entity;  
globals.h - здесь объявляются и определяются глобальные переменные;  
grant.h - интерфейс класса Grant;  
HealthBonus.h - интерфейс класса HealthBonus;  
hero.h - интерфейс класса Hero;  
Magma.h - интерфейс класса Magma;  
MapCreator.h – объявление функций создания карты;  
TowerWithArcher.h - интерфейс класса TowerArch;

Единицы компиляции:  
arrow.cpp – реализация методов класса Arrow;  
DamageBonus. cpp – реализация методов класса DamageBonus;  
Entity.cpp – реализация методов класса Entity;  
grant.cpp – реализация методов класса Grant;  
HealthBonus.cpp - реализация методов класса HealthBonus;  
hero.cpp – реализация методов класса Hero;  
Magma.cpp – реализация методов класса Magma;  
MapCreator.cpp – определение функций создания карты;  
TowerWithArcher.cpp – реализация методов класса TowerArch;  
main.cpp – главный файл, здесь объявляется переменная hero класса Hero, также создаются контейнер list под класс Entity и итератор для него, также создается контейнер list для класса Arrow и итератор для него. Имеется цикл, действующий пока окно открыто, в нем определяется изменение положения карты, в зависимости от направления движения игрока, вывод карты на экран, добавление различных врагов и бонусов в list класса Entity, обработка пересечений hero с бонусами и стрелами, обработка атаки героем противника, обработка атаки противниками героя, вывод бонусов, героя, стрел и врагов на экран, удаление стрел, если они попали в героя или улетели за пределы карты. Удаление бонусов, если они подобраны героем или вышли за пределы карты. Удаление врагов, если они вышли за пределы карты. Вывод текущего количества очков и жизней. После цикла происходит очистка памяти.

ЗАКЛЮЧЕНИЕ  
Среда разработки: Microsoft Visual Studio  
Язык программирования: C++  
Применялась ли система контроля версий Git: да  
Все прецеденты реализованы  
Сбоев и зависаний не наблюдается  
Наблюдаются небольшие баги, связанные с появление лавы поверх башни  
Использовался принцип раздельной компиляции  
Реализована очистка памяти  
Отсутствуют неиспользуемые переменные  
Возможна избыточность, но мы о ней тогда не знаем  
Приложена диаграмма классов UML  
Цель достигнута  
Рабочую последнюю версию игры можно взять тут: https://github.com/Ifksitovec/LastChel.git

ДИАГРАММА КЛАССОВ UML  


ЛИСТИНГ ПРОГРАММЫ

//arrow.h

#include "Entity.h"

#ifndef \_\_ARROW\_H\_\_

#define \_\_ARROW\_H\_\_

class Arrow : public Entity

{

public:

Arrow(float, float);

void update(float,float);

void control();

void SetHp(int, Gtype);

bool RadiusDamage(float,float);

};

#endif

-----------------------------------------------------------------------------------------

//DamageBonus.h

#include "Entity.h"

#ifndef \_\_DAMAGEBONUS\_H\_\_

#define \_\_DAMAGEBONUS\_H\_\_

class DamageBonus : public Entity

{

public:

DamageBonus(float, float);

void update(float,float);

void control();

void SetHp(int, Gtype);

bool RadiusDamage(float,float);

};

#endif

-----------------------------------------------------------------------------------------

//Entity.h

#include "SFML/Graphics.hpp"

#include "globals.h"

using namespace sf;

#ifndef \_\_CLASSESS\_H\_\_

#define \_\_CLASSESS\_H\_\_

class Entity

{

protected:

Gstate PrevState;

int ShootTime;

int hp, score;

float CurrentFrame;

Clock time;

bool life;

float speed, dx, dy, PosX, PosY;

Gstate state;

Gtype type;

Texture texture;

Image image;

Sprite sprite;

int damage;

float N; // счетчик атаки

public:

Gtype GetType();

FloatRect GetRect();

int GetDamage();

void ChangePos(float,float);

void SetStay();

Gstate GetState();

Sprite GetImage();

virtual bool RadiusDamage(float,float) = 0;

virtual void update(float,float) = 0;

virtual void control() = 0;

virtual void SetHp(int, Gtype) = 0;

int GetHp();

float GetPosX();

float GetPosY();

bool GetLife();

void SetLife(bool);

void SetScore(int);

int GetScore();

float GetN(); // получение значения счетчика атаки

void SetN(float); // изменение счетчика атаки

bool ShootDelay();

};

#endif

-----------------------------------------------------------------------------------------

//globals.h

#include "SFML/Graphics.hpp"

#ifndef \_\_GLOBALS\_H\_\_

#define \_\_GLOBALS\_H\_\_

enum Gstate { west, NorthWest, north , NorthEast, east, SouthEast, south, SouthWest, stay, attack, block};

enum Gtype { pers, arrow, BonusHp, BonusDam, tow, magma };

const int size = 800;

const int SizeImg = 50;

const int SizeImgArr = 25;

const int SizeImgTown = SizeImg + 20;

const int HeightMap = 36;

const int SizeImgMap = 25;

extern char TileMap[HeightMap][HeightMap];

const int delay = 2400;

#endif

-----------------------------------------------------------------------------------------

//grant.h

#include "Entity.h"

#ifndef \_\_GRANT\_H\_\_

#define \_\_GRANT\_H\_\_

class Grant : public Entity

{

public:

Grant(float, float);

void update(float,float);

void control();

void SetHp(int, Gtype);

bool RadiusDamage(float,float);

};

#endif

-----------------------------------------------------------------------------------------

//HealthBonus.h

#include "Entity.h"

#ifndef \_\_HAELTHBONUS\_H\_\_

#define \_\_HEALTHBONUS\_H\_\_

class HealthBonus : public Entity

{

public:

HealthBonus(float, float);

void update(float,float);

void control();

void SetHp(int, Gtype);

bool RadiusDamage(float,float);

};

#endif

-----------------------------------------------------------------------------------------

//hero.h

#include "Entity.h"

#ifndef \_\_HERO\_H\_\_

#define \_\_HERO\_H\_\_

class Hero : public Entity

{

protected:

int TimeAction;

Clock time2;

public:

Hero();

void update(float,float);

void control();

void SetHp(int, Gtype);

bool RadiusDamage(float,float);

float GetSpeed();

bool TimeOfAction();

void StartTime2();

int GetTimeAction();

};

#endif

-----------------------------------------------------------------------------------------

//Magma.h

#include "Entity.h"

#ifndef \_\_MAGMA\_H\_\_

#define \_\_MAGMA\_H\_\_

class Magma : public Entity

{

public:

Magma(float, float);

void update(float,float);

void control();

void SetHp(int, Gtype);

bool RadiusDamage(float,float);

};

#endif

-----------------------------------------------------------------------------------------

//MapCreator.h

#include "globals.h"

#ifndef \_\_MAPCREATOR\_H\_\_

#define \_\_MAPCREATOR\_H\_\_

void RandomMapGenerate();

void UpdateMap(int);

#endif

-----------------------------------------------------------------------------------------

//TowerWithArcher.h

#include "Entity.h"

#ifndef \_\_TOWERARCH\_H\_\_

#define \_\_TOWERARCH\_H\_\_

class TowerArch : public Entity

{

public:

TowerArch(float, float);

void update(float,float);

void control();

void SetHp(int, Gtype);

bool RadiusDamage(float,float);

};

#endif

-----------------------------------------------------------------------------------------

//arrow.cpp

#include "arrow.h"

#include <cmath>

using namespace std;

Arrow::Arrow(float x, float y)

{

PrevState = east;//направляющее положение

hp = 50;

PosX = x;

PosY = y;

speed = 0.25;

type = arrow;

CurrentFrame = 0;

life = true;

damage = -20;

texture.loadFromFile("images/arm.png");

sprite.setTexture(texture);

sprite.setPosition(PosX, PosY);

if (((int)(x - size/2) > 0) && ((int)(y - size/2) > 0)) //северо-запад

{

state = NorthWest;

sprite.setTextureRect(IntRect(279, 12, SizeImgArr, SizeImgArr));

}

if (((int)(x - size/2) > 0) && ((int)(y - size/2) < 0)) //юго-запад

{

state = SouthWest;

sprite.setTextureRect(IntRect(193, 14, SizeImgArr, SizeImgArr));

}

if (((int)(x - size/2) < 0) && ((int)(y - size/2) > 0)) //северо-восток

{

state = NorthEast;

sprite.setTextureRect(IntRect(40, 14, SizeImgArr, SizeImgArr));

}

if (((int)(x - size/2) < 0) && ((int)(y - size/2) < 0)) //юго-восток

{

state = SouthEast;

sprite.setTextureRect(IntRect(120, 17, SizeImgArr, SizeImgArr));

}

if (((int)(x - size/2) == 0) && ((int)(y - size/2) > 0)) //север

{

state = north;

sprite.setTextureRect(IntRect(8, 10, SizeImgArr - 5, SizeImgArr + 5));

}

if (((int)(x - size/2) == 0) && ((int)(y - size/2) < 0)) //юг

{

state = south;

sprite.setTextureRect(IntRect(158, 13, SizeImgArr - 5, SizeImgArr + 5));

}

if (((int)(x - size/2) < 0) && ((int)(y - size/2) == 0)) //восток

{

state = east;

sprite.setTextureRect(IntRect(74, 18, SizeImgArr + 5, SizeImgArr - 5));

}

if (((int)(x - size/2) > 0) && ((int)(y - size/2) == 0)) //запад

{

state = west;

sprite.setTextureRect(IntRect(233, 15, SizeImgArr + 5, SizeImgArr - 5));

}

switch (state)

{

case north:

{

dx = 0;

dy = -speed;

break;

}

case south:

{

dx = 0;

dy = speed;

break;

}

case west:

{

dx = -speed;

dy = 0;

break;

}

case east:

{

dx = speed;

dy = 0;

break;

}

case NorthWest:

{

dx = -speed \* cos(atan((size/2 - y)/(size/2 - x)));

dy = -speed \* sin(atan((size/2 - y)/(size/2 - x)));

break;

}

case NorthEast:

{

dx = speed \* cos(atan((size/2 - y)/(x - size/2)));

dy = -speed \* sin(atan((size/2 - y)/(x - size/2)));

break;

}

case SouthEast:

{

dx = speed \* cos(atan((y - size/2)/(x - size/2)));

dy = speed \* sin(atan((y - size/2)/(x - size/2)));

break;

}

case SouthWest:

{

dx = -speed \* cos(atan((y - size/2)/(size/2 - x)));

dy = speed \* sin(atan((y - size/2)/(size/2 - x)));

break;

}

}

}

void Arrow::SetHp(int h, Gtype t)

{

hp = hp;

}

bool Arrow::RadiusDamage(float x,float y)

{

return true;

}

void Arrow::control()

{

state = state;

}

void Arrow::update(float time1,float n1)

{

if (life)

{

if ((int)n1 == 0)

{

control();

switch (state)

{

case north:

{

sprite.setTextureRect(IntRect(8, 10, SizeImgArr - 5, SizeImgArr + 5));

break;

}

case south:

{

sprite.setTextureRect(IntRect(158, 13, SizeImgArr - 5, SizeImgArr + 5));

break;

}

case east:

{

sprite.setTextureRect(IntRect(74, 18, SizeImgArr + 5, SizeImgArr - 5));

break;

}

case west:

{

sprite.setTextureRect(IntRect(233, 15, SizeImgArr + 5, SizeImgArr - 5));

break;

}

case NorthWest:

{

sprite.setTextureRect(IntRect(279, 12, SizeImgArr, SizeImgArr));

break;

}

case NorthEast:

{

sprite.setTextureRect(IntRect(40, 14, SizeImgArr, SizeImgArr));

break;

}

case SouthEast:

{

sprite.setTextureRect(IntRect(120, 17, SizeImgArr, SizeImgArr));

break;

}

case SouthWest:

{

sprite.setTextureRect(IntRect(194, 13, SizeImgArr, SizeImgArr));

break;

}

}

}

PosX += dx \* time1;

PosY += dy \* time1;

}

sprite.setPosition(PosX, PosY);

}

-----------------------------------------------------------------------------------------

//DamageBonus.cpp

#include "DamageBonus.h"

DamageBonus::DamageBonus(float x, float y)

{

PrevState = east;//направляющее положение

state = stay; // начальное положение

hp = 300;

PosX = x;

PosY = y;

speed = 0;

type = BonusDam;

CurrentFrame = 0;

life = true;

damage = 0;

texture.loadFromFile("images/map.png");

sprite.setTexture(texture);

sprite.setPosition(PosX, PosY);

sprite.setTextureRect(IntRect(100, 200, SizeImgMap, SizeImgMap));

}

bool DamageBonus::RadiusDamage(float x,float y)

{

return true;

}

void DamageBonus::SetHp(int h, Gtype t)

{

hp = hp;

}

void DamageBonus::control()

{

state = stay;

}

void DamageBonus::update(float time,float n1)

{

if (life)

{

control();

sprite.setTextureRect(IntRect(100, 200, SizeImgMap, SizeImgMap));

}

sprite.setPosition(PosX, PosY);

}

-----------------------------------------------------------------------------------------

//Entity.cpp

#include "Entity.h"

#include <iostream>

#include <cmath>

void Entity::SetN(float time)

{

if (N > 4) N = 0;

else N += 0.01\*time;

}

float Entity::GetN()

{

return N;

}

bool Entity::ShootDelay()

{

ShootTime = time.getElapsedTime().asMilliseconds();

if (ShootTime >= delay)

{

time.restart();

return true;

}

else

return false;

}

void Entity::SetStay()

{

state = stay;

}

Gstate Entity::GetState()

{

return state;

}

Sprite Entity::GetImage()

{

return sprite;

}

bool Entity::GetLife()

{

return life;

}

void Entity::SetLife(bool l)

{

life = l;

}

void Entity::SetScore(int s)

{

score += s;

}

int Entity::GetScore()

{

return score;

}

int Entity::GetHp()

{

return hp;

}

//получение позиции по Х персонажа

float Entity::GetPosX()

{

return PosX;

}

//получение позиции по У персонажа

float Entity::GetPosY()

{

return PosY;

}

void Entity::ChangePos(float x, float y)

{

PosX += x;

PosY += y;

}

int Entity::GetDamage()

{

return damage;

}

FloatRect Entity::GetRect()

{

if (type == pers)

{

FloatRect FR(PosX, PosY, SizeImg, SizeImg);

return FR;

}

else

if (type == arrow)

{

FloatRect FR(PosX, PosY, SizeImgArr, SizeImgArr);

return FR;

}

else

if ((type == BonusHp) || (type == BonusDam) || (type == magma))

{

FloatRect FR(PosX, PosY, SizeImgMap, SizeImgMap);

return FR;

}

}

Gtype Entity::GetType()

{

return type;

}

-----------------------------------------------------------------------------------------

//grant.cpp

#include "grant.h"

Grant::Grant(float x, float y)

{

PrevState = east;//направляющее положение

state = stay; // начальное положение

hp = 150;

PosX = x;

PosY = y;

speed = 0.05;

type = pers;

CurrentFrame = 0;

life = true;

damage = -20;

N = 0;

texture.loadFromFile("images/grant.png");

sprite.setTexture(texture);

sprite.setPosition(PosX, PosY);

sprite.setTextureRect(IntRect(36, 176, SizeImg - 15, SizeImg));

}

bool Grant::RadiusDamage(float x,float y)

{

float dposX = x, dposY = y;

int s = SizeImg - 20;

if (life)

{

switch (PrevState)

{

case north:

{

if ((PosY - ((size + s)/2) ) < 15) return true;

else false;

break;

}

case west:

{

if ((PosX - ((size + s)/2) ) < 15) return true;

else false;

break;

}

case east:

{

if ((((size + 3 \* s)/2) - PosX) < 15) return true;

else false;

break;

}

case south:

{

if ((((size + 3 \* s)/2) - PosY) < 15) return true;

else false;

break;

}

case NorthEast:

{

if (((PosY - ((size + s)/2) ) < 15) && ((((size + 3 \* s)/2) - PosX) < 15)) return true;

else false;

break;

}

case NorthWest:

{

if (((PosX - ((size + s)/2) ) < 15) && ((PosY - ((size + s)/2) ) < 15)) return true;

else false;

break;

}

case SouthEast:

{

if (((((size + 3 \* s)/2) - PosX) < 15) && ((((size + 3 \* s)/2) - PosY) < 15)) return true;

else false;

break;

}

case SouthWest:

{

if (((((size + 3 \* s)/2) - PosY) < 15) && ((PosX - ((size + s)/2) ) < 15)) return true;

else false;

break;

}

}

}

}

void Grant::SetHp(int h, Gtype t)

{

hp += h;

}

//определеение направления движения

void Grant::control()

{

float dposX = PosX - size/2, dposY = PosY - size/ 2;

int s = SizeImg - 20;

//Если объект достиг героя и стоит

if (((state == attack) || (state == stay)) && (PosY <= (size + s)/2) && (PosY >= (size - 3 \* s)/2) && (PosX <= (size + s)/2) && (PosX >= (size - 3 \* s)/2))

{

state = attack;

}

else

{

if ((dposX > 0) && (dposY > 0) && (PosX > (size + s)/2) && (PosY > (size + s)/2))// если объект находится в правом нижнем углу

{

if (dposX > dposY) state = north;

if (dposX < dposY) state = west;

if (dposX == dposY) state = NorthWest;

}

if ((dposX > 0) && (dposY < 0) && (PosX > (size + s)/2) && (PosY < (size - 3 \* s)/2))// если объект находится в правом верхнем углу

{

if (dposX > -dposY) state = south;

if (dposX < -dposY) state = west;

if (dposX == -dposY) state = SouthWest;

}

if ((dposX < 0) && (dposY > 0) && (PosX < (size - 3 \* s)/2) && (PosY > (size + s)/2))// если объект находится в левом нижнем углу

{

if (-dposX > dposY) state = north;

if (-dposX < dposY) state = east;

if (-dposX == dposY) state = NorthEast;

}

if ((dposX < 0) && (dposY < 0) && (PosX < (size - 3 \* s)/2) && (PosY < (size - 3 \* s)/2))// если объект находится в правом нижнем углу

{

if (-dposX > -dposY) state = south;

if (-dposX < -dposY) state = east;

if (-dposX == -dposY) state = SouthEast;

}

if ((PosX <= (size + s)/2) && (PosX >= (size - 3 \* s)/2))// если объект находится сверху или снизу от героя

{

if (dposY > 0) state = north;

if (dposY < 0) state = south;

}

if ((PosY <= (size + s)/2) && (PosY >= (size - 3 \* s)/2))// если объект находится справа или слева от героя

{

if (dposX > 0) state = west;

if (dposX < 0) state = east;

}

if ((PosY <= (size + s)/2) && (PosY >= (size - 3 \* s)/2) && (PosX <= (size + s)/2) && (PosX >= (size - 3 \* s)/2)) state = stay; // если объект достиг героя

}

}

//изменение движения

void Grant::update(float time,float n1)

{

if (life)

{

control();

if ((int)n1 == 0)

{

switch (state)

{

case west://движение налево

{

dx = -speed;

dy = 0;

CurrentFrame += 0.01 \* time;

if (CurrentFrame > 4) CurrentFrame -= 4;

sprite.setTextureRect(IntRect(14 + 60 \* int(CurrentFrame), 381, SizeImg +10, SizeImg));

PrevState = west;

break;

}

case north://движение вверх

{

dx = 0;

dy = -speed;

CurrentFrame += 0.01 \* time;

if (CurrentFrame > 4) CurrentFrame -= 4;

sprite.setTextureRect(IntRect(60 \* int(CurrentFrame), 0, SizeImg +10, SizeImg));

PrevState = north;

break;

}

case east://движение направо

{

dx = speed;

dy = 0;

CurrentFrame += 0.01 \* time;

if (CurrentFrame > 4) CurrentFrame -= 4;

sprite.setTextureRect(IntRect(60 \* int(CurrentFrame), 125, SizeImg + 10, SizeImg));

PrevState = east;

break;

}

case south://движение вниз

{

dx = 0;

dy = speed;

CurrentFrame += 0.01 \* time;

if (CurrentFrame > 4) CurrentFrame -= 4;

sprite.setTextureRect(IntRect(60 \* int(CurrentFrame), 252, SizeImg + 10, SizeImg));

PrevState = south;

break;

}

case NorthWest://движение вверх-налево

{

dx = -speed;

dy = -speed;

CurrentFrame += 0.01 \* time;

if (CurrentFrame > 4) CurrentFrame -= 4;

sprite.setTextureRect(IntRect(6 + 61 \* int(CurrentFrame), 446, SizeImg + 11, SizeImg));

PrevState = NorthWest;

break;

}

case NorthEast://движение вверх-направо

{

dx = speed;

dy = -speed;

CurrentFrame += 0.01 \* time;

if (CurrentFrame > 4) CurrentFrame -= 4;

sprite.setTextureRect(IntRect(61 \* int(CurrentFrame), 62, SizeImg + 11, SizeImg));

PrevState = NorthEast;

break;

}

case SouthEast://движение вниз-направо

{

dx = speed;

dy = speed;

CurrentFrame += 0.01 \* time;

if (CurrentFrame > 4) CurrentFrame -= 4;

sprite.setTextureRect(IntRect(60 \* int(CurrentFrame), 188, SizeImg + 10, SizeImg));

PrevState = SouthEast;

break;

}

case SouthWest://движение вниз-налево

{

dx = -speed;

dy = speed;

CurrentFrame += 0.01 \* time;

if (CurrentFrame > 4) CurrentFrame -= 4;

sprite.setTextureRect(IntRect(11 + 60 \* int(CurrentFrame), 315, SizeImg + 10, SizeImg));

PrevState = SouthWest;

break;

}

case stay://положение стоять в зависимости от того какое до этого было положение

{

CurrentFrame = 0;

switch (PrevState)

{

case north:

{

sprite.setTextureRect(IntRect(60 \* int(CurrentFrame), 0, SizeImg +10, SizeImg));

break;

}

case east:

{

sprite.setTextureRect(IntRect(60 \* int(CurrentFrame), 125, SizeImg + 10, SizeImg));

break;

}

case west:

{

sprite.setTextureRect(IntRect(14 + 60 \* int(CurrentFrame), 381, SizeImg +10, SizeImg));

break;

}

case south:

{

sprite.setTextureRect(IntRect(60 \* int(CurrentFrame), 252, SizeImg + 10, SizeImg));

break;

}

case NorthWest:

{

sprite.setTextureRect(IntRect(6 + 61 \* int(CurrentFrame), 446, SizeImg + 11, SizeImg));

break;

}

case NorthEast:

{

sprite.setTextureRect(IntRect(61 \* int(CurrentFrame), 62, SizeImg + 11, SizeImg));

break;

}

case SouthWest:

{

sprite.setTextureRect(IntRect(11 + 60 \* int(CurrentFrame), 315, SizeImg + 10, SizeImg));

break;

}

case SouthEast:

{

sprite.setTextureRect(IntRect(60 \* int(CurrentFrame), 188, SizeImg + 10, SizeImg));

break;

}

}

dx = 0;

dy = 0;

}

}

}

else //отображение атаки

{

switch (PrevState)

{

case north:

{

sprite.setTextureRect(IntRect(318 + 60 \* int(n1), 0, SizeImg + 10, SizeImg));

if ((int)n1 == 4) sprite.setTextureRect(IntRect(0, 0, SizeImg +10, SizeImg));

break;

}

case east:

{

sprite.setTextureRect(IntRect(314 + 63 \* int(n1), 124, SizeImg + 13, SizeImg + 10));

if ((int)n1 == 4) sprite.setTextureRect(IntRect(0, 125, SizeImg + 10, SizeImg));

break;

}

case west:

{

sprite.setTextureRect(IntRect(327 + 60 \* int(n1), 380, SizeImg +10, SizeImg + 10));

if ((int)n1 == 4) sprite.setTextureRect(IntRect(14, 381, SizeImg +10, SizeImg));

break;

}

case south:

{

sprite.setTextureRect(IntRect(310 + 63 \* int(n1), 251, SizeImg + 13, SizeImg + 10));

if ((int)n1 == 4) sprite.setTextureRect(IntRect(0, 252, SizeImg + 10, SizeImg));

break;

}

case NorthWest:

{

sprite.setTextureRect(IntRect(315 + 64 \* int(n1), 445, SizeImg + 14, SizeImg));

if ((int)n1 == 4) sprite.setTextureRect(IntRect(6, 446, SizeImg + 11, SizeImg));

break;

}

case NorthEast:

{

sprite.setTextureRect(IntRect(315 + 62 \* int(n1), 61, SizeImg + 12, SizeImg));

if ((int)n1 == 4) sprite.setTextureRect(IntRect(0, 62, SizeImg + 11, SizeImg));

break;

}

case SouthWest:

{

sprite.setTextureRect(IntRect(329 + 60 \* int(n1), 315, SizeImg + 10, SizeImg + 10));

if ((int)n1 == 4) sprite.setTextureRect(IntRect(11, 315, SizeImg + 10, SizeImg));

break;

}

case SouthEast:

{

sprite.setTextureRect(IntRect(308 + 65 \* int(n1), 187, SizeImg + 15, SizeImg));

if ((int)n1 == 4) sprite.setTextureRect(IntRect(0, 188, SizeImg + 10, SizeImg));

break;

}

}

dx = 0;

dy = 0;

}

PosX += dx \* time;

PosY += dy \* time;

//state = stay;

if (hp <= 0)

{

life = false;

sprite.setTextureRect(IntRect(699, 254, SizeImg + 10, SizeImg + 3));

dx = 0;

dy = 0;

}

}

sprite.setPosition(PosX, PosY);

}

-----------------------------------------------------------------------------------------

//HealthBonus.cpp

#include "HealthBonus.h"

HealthBonus::HealthBonus(float x, float y)

{

PrevState = east;//направляющее положение

state = stay; // начальное положение

hp = 300;

PosX = x;

PosY = y;

speed = 0;

type = BonusHp;

CurrentFrame = 0;

life = true;

damage = 200;

texture.loadFromFile("images/map.png");

sprite.setTexture(texture);

sprite.setPosition(PosX, PosY);

sprite.setTextureRect(IntRect(50, 175, SizeImgMap, SizeImgMap));

}

bool HealthBonus::RadiusDamage(float x,float y)

{

return true;

}

void HealthBonus::SetHp(int h, Gtype)

{

hp = hp;

}

void HealthBonus::control()

{

state = stay;

}

void HealthBonus::update(float time,float n1)

{

if (life)

{

control();

sprite.setTextureRect(IntRect(50, 175, SizeImgMap, SizeImgMap));

}

sprite.setPosition(PosX, PosY);

}

-----------------------------------------------------------------------------------------

//Hero.cpp

#include "Hero.h"

Hero::Hero()

{

PrevState = east;//направляющие положение

state = stay; // начальное положение

hp = 1000;

PosX = (size - SizeImg) / 2;

PosY = (size - SizeImg) / 2;

speed = 0.1;

type = pers;

score = 0;

CurrentFrame = 0;

life = true;

damage = -50;

texture.loadFromFile("images/footman.png");

sprite.setTexture(texture);

sprite.setPosition(PosX, PosY);

sprite.setTextureRect(IntRect(0, 127, SizeImg, SizeImg));

}

int Hero::GetTimeAction()

{

return TimeAction;

}

void Hero::StartTime2()

{

time2.restart();

}

bool Hero::TimeOfAction()

{

TimeAction = 10 \* delay - time2.getElapsedTime().asMilliseconds();

if (TimeAction > 0) return true;

else

{

if (TimeAction < 0) return false;

}

}

float Hero::GetSpeed()

{

return speed;

}

void Hero::SetHp(int h, Gtype t)

{

if ((state != block) || (t == magma)) hp += h;

if (hp < 0) hp = 0;

}

bool Hero::RadiusDamage(float x, float y)// определение находится ли противник в зоне атаки

{

switch (PrevState)

{

case north:

{

if ((x >= PosX - SizeImg) && (x <= PosX + 2 \* SizeImg) && ((PosY - SizeImg/2 - y) < 25) & ((PosY - SizeImg/2 - y) > -25)) return true;

else return false;

break;

}

case west:

{

if ((y >= PosY - SizeImg) && (y <= PosY + 2 \* SizeImg) && ((PosX - SizeImg/2 - x) < 25) && ((PosX - SizeImg/2 - x) > -25)) return true;

else return false;

break;

}

case east:

{

if ((y >= PosY - SizeImg) && (y <= PosY + 2 \* SizeImg) && ((x - PosX - SizeImg/2) < 25) && ((x - PosX - SizeImg/2) > -25)) return true;

else return false;

break;

}

case south:

{

if ((x >= PosX - SizeImg) && (x <= PosX + 2 \* SizeImg) && ((y - PosY - SizeImg/2) < 25) && ((y - PosY - SizeImg/2) > -25)) return true;

else return false;

break;

}

case NorthEast:

{

if (((x - PosX - SizeImg/2) < 25) && ((PosY - SizeImg/2 - y) < 25) && ((x - PosX - SizeImg/2) > -25) && ((PosY - SizeImg/2 - y) > -25)) return true;

else return false;

break;

}

case NorthWest:

{

if (((PosY - SizeImg/2 - y) < 25) && ((PosX - SizeImg/2 - x) < 25) && ((PosY - SizeImg/2 - y) > -25) && ((PosX - SizeImg/2 - x) > -25)) return true;

else return false;

break;

}

case SouthEast:

{

if (((y - PosY - SizeImg/2) < 25) && ((x - PosX - SizeImg/2) < 25) && ((y - PosY - SizeImg/2) > -25) && ((x - PosX - SizeImg/2) > -25)) return true;

else return false;

break;

}

case SouthWest:

{

if (((PosX - SizeImg/2/ - x) < 25) && ((y - PosY - SizeImg/2) < 25) && ((PosX - SizeImg/2 - x) > -25) && ((y - PosY - SizeImg/2) > -25)) return true;

else return false;

break;

}

default:

{

return false;

}

}

}

//действия по нажатию клавиш

void Hero::control()

{

if (Keyboard::isKeyPressed(Keyboard::Left))

{

state = west;

}

if (Keyboard::isKeyPressed(Keyboard::Right))

{

state = east;

}

if (Keyboard::isKeyPressed(Keyboard::Up))

{

state = north;

}

if (Keyboard::isKeyPressed(Keyboard::Down))

{

state = south;

}

if ((Keyboard::isKeyPressed(Keyboard::Down)) && (Keyboard::isKeyPressed(Keyboard::Left)))

{

state = SouthWest;

}

if ((Keyboard::isKeyPressed(Keyboard::Down)) && (Keyboard::isKeyPressed(Keyboard::Right)))

{

state = SouthEast;

}

if ((Keyboard::isKeyPressed(Keyboard::Up)) && (Keyboard::isKeyPressed(Keyboard::Left)))

{

state = NorthWest;

}

if ((Keyboard::isKeyPressed(Keyboard::Up)) && (Keyboard::isKeyPressed(Keyboard::Right)))

{

state = NorthEast;

}

if ((Keyboard::isKeyPressed(Keyboard::Left)) && (Keyboard::isKeyPressed(Keyboard::Right)))

{

state = stay;

}

if ((Keyboard::isKeyPressed(Keyboard::Up)) && (Keyboard::isKeyPressed(Keyboard::Down)))

{

state = stay;

}

if (Keyboard::isKeyPressed(Keyboard::Space))

{

state = attack;

}

if (Keyboard::isKeyPressed(Keyboard::LAlt))

{

state = block;

}

}

//Изменение картинки персонажа

void Hero::update(float time, float n)

{

if (life)

{

control();

if (n == 0)

{

switch (state)

{

case east: //Движение вправо

CurrentFrame +=0.01\*time;

if (CurrentFrame > 5) CurrentFrame -= 5;

switch (int(CurrentFrame))

{

case 1:

{

sprite.setTextureRect(IntRect(0, 122, SizeImg, SizeImg));

break;

}

case 2:

{

sprite.setTextureRect(IntRect(60, 122, SizeImg, SizeImg));

break;

}

case 3:

{

sprite.setTextureRect(IntRect(122, 122, SizeImg, SizeImg));

break;

}

case 4:

{

sprite.setTextureRect(IntRect(185, 122, SizeImg, SizeImg));

break;

}

case 5:

{

sprite.setTextureRect(IntRect(248, 122, SizeImg, SizeImg));

break;

}

}

PrevState = east;

break;

case west: //Движение влево

CurrentFrame +=0.01\*time;

if (CurrentFrame > 5) CurrentFrame -= 5;

switch (int(CurrentFrame))

{

case 1:

{

sprite.setTextureRect(IntRect(5, 376, SizeImg, SizeImg));

break;

}

case 2:

{

sprite.setTextureRect(IntRect(65, 378, SizeImg, SizeImg));

break;

}

case 3:

{

sprite.setTextureRect(IntRect(127, 378, SizeImg, SizeImg));

break;

}

case 4:

{

sprite.setTextureRect(IntRect(197, 378, SizeImg, SizeImg));

break;

}

case 5:

{

sprite.setTextureRect(IntRect(257, 378, SizeImg, SizeImg));

break;

}

}

PrevState = west;

break;

case north: //Движение вверх

CurrentFrame +=0.01\*time;

if (CurrentFrame > 5) CurrentFrame -= 5;

switch (int(CurrentFrame))

{

case 1:

{

sprite.setTextureRect(IntRect(0, 0, SizeImg, SizeImg));

break;

}

case 2:

{

sprite.setTextureRect(IntRect(58, 3, SizeImg, SizeImg));

break;

}

case 3:

{

sprite.setTextureRect(IntRect(117, 3, SizeImg, SizeImg));

break;

}

case 4:

{

sprite.setTextureRect(IntRect(183, 0, SizeImg, SizeImg));

break;

}

case 5:

{

sprite.setTextureRect(IntRect(248, 0, SizeImg, SizeImg));

break;

}

}

PrevState = north;

break;

case south: //Движение вниз

CurrentFrame +=0.01\*time;

if (CurrentFrame > 5) CurrentFrame -= 5;

switch (int(CurrentFrame))

{

case 1:

{

sprite.setTextureRect(IntRect(0, 248, SizeImg, SizeImg));

break;

}

case 2:

{

sprite.setTextureRect(IntRect(58, 250, SizeImg, SizeImg));

break;

}

case 3:

{

sprite.setTextureRect(IntRect(117, 251, SizeImg, SizeImg));

break;

}

case 4:

{

sprite.setTextureRect(IntRect(185, 252, SizeImg, SizeImg));

break;

}

case 5:

{

sprite.setTextureRect(IntRect(250, 251, SizeImg, SizeImg));

break;

}

}

PrevState = south;

break;

case NorthEast: //Движение вверх-вправо

CurrentFrame +=0.01\*time;

if (CurrentFrame > 5) CurrentFrame -= 5;

switch (int(CurrentFrame))

{

case 1:

{

sprite.setTextureRect(IntRect(0, 58, SizeImg, SizeImg));

break;

}

case 2:

{

sprite.setTextureRect(IntRect(57, 59, SizeImg, SizeImg));

break;

}

case 3:

{

sprite.setTextureRect(IntRect(121, 61, SizeImg, SizeImg));

break;

}

case 4:

{

sprite.setTextureRect(IntRect(183, 57, SizeImg, SizeImg));

break;

}

case 5:

{

sprite.setTextureRect(IntRect(248, 59, SizeImg, SizeImg));

break;

}

}

PrevState = NorthEast;

break;

case NorthWest: //Движение вверх-влево

CurrentFrame +=0.01\*time;

if (CurrentFrame > 5) CurrentFrame -= 5;

switch (int(CurrentFrame))

{

case 1:

{

sprite.setTextureRect(IntRect(5, 442, SizeImg, SizeImg));

break;

}

case 2:

{

sprite.setTextureRect(IntRect(62, 443, SizeImg, SizeImg));

break;

}

case 3:

{

sprite.setTextureRect(IntRect(132, 443, SizeImg, SizeImg));

break;

}

case 4:

{

sprite.setTextureRect(IntRect(196, 443, SizeImg, SizeImg));

break;

}

case 5:

{

sprite.setTextureRect(IntRect(262, 439, SizeImg, SizeImg));

break;

}

}

PrevState = NorthWest;

break;

case SouthEast: //Движение вниз-вправо

CurrentFrame +=0.01\*time;

if (CurrentFrame > 5) CurrentFrame -= 5;

switch (int(CurrentFrame))

{

case 1:

{

sprite.setTextureRect(IntRect(0, 185, SizeImg, SizeImg));

break;

}

case 2:

{

sprite.setTextureRect(IntRect(57, 187, SizeImg, SizeImg));

break;

}

case 3:

{

sprite.setTextureRect(IntRect(120, 185, SizeImg, SizeImg));

break;

}

case 4:

{

sprite.setTextureRect(IntRect(185, 185, SizeImg, SizeImg));

break;

}

case 5:

{

sprite.setTextureRect(IntRect(249, 185, SizeImg, SizeImg));

break;

}

}

PrevState = SouthEast;

break;

case SouthWest: //Движение вниз-влево

CurrentFrame +=0.01\*time;

if (CurrentFrame > 5) CurrentFrame -= 5;

switch (int(CurrentFrame))

{

case 1:

{

sprite.setTextureRect(IntRect(0, 313, SizeImg, SizeImg));

break;

}

case 2:

{

sprite.setTextureRect(IntRect(59, 315, SizeImg, SizeImg));

break;

}

case 3:

{

sprite.setTextureRect(IntRect(127, 315, SizeImg, SizeImg));

break;

}

case 4:

{

sprite.setTextureRect(IntRect(191, 313, SizeImg, SizeImg));

break;

}

case 5:

{

sprite.setTextureRect(IntRect(256, 313, SizeImg, SizeImg));

break;

}

}

PrevState = SouthWest;

break;

case block: // блокировка

{

switch (PrevState)

{

case north:

{

sprite.setTextureRect(IntRect(377, 0, SizeImg, SizeImg));

break;

}

case west:

{

sprite.setTextureRect(IntRect(368, 380, SizeImg + 10, SizeImg));

break;

}

case east:

{

sprite.setTextureRect(IntRect(383, 125, SizeImg + 10, SizeImg));

break;

}

case south:

{

sprite.setTextureRect(IntRect(330, 318, SizeImg - 10, SizeImg));

break;

}

case NorthWest:

{

sprite.setTextureRect(IntRect(374, 446, SizeImg, SizeImg));

break;

}

case NorthEast:

{

sprite.setTextureRect(IntRect(385, 61, SizeImg, SizeImg));

break;

}

case SouthWest:

{

sprite.setTextureRect(IntRect(315, 251, SizeImg, SizeImg));

break;

}

case SouthEast:

{

sprite.setTextureRect(IntRect(381, 192, SizeImg - 10, SizeImg));

break;

}

}

break;

}

}

}

else

{

switch (PrevState )

{

case east://изображение атаки при направлении направо

{

switch (int(n))

{

case 0:

{

sprite.setTextureRect(IntRect(314, 127, SizeImg, SizeImg));

break;

}

case 1:

{

sprite.setTextureRect(IntRect(384, 126, SizeImg+10, SizeImg));

break;

}

case 2:

{

sprite.setTextureRect(IntRect(443, 126, SizeImg, SizeImg+5));

break;

}

case 3:

{

sprite.setTextureRect(IntRect(510, 126, SizeImg+10, SizeImg));

break;

}

case 4:

{

sprite.setTextureRect(IntRect(0, 120, SizeImg, SizeImg));

break;

}

}

break;

}

case west://изображение атаки при направлении налево

{

switch (int(n))

{

case 0:

{

sprite.setTextureRect(IntRect(327, 384, SizeImg - 10,SizeImg));

break;

}

case 1:

{

sprite.setTextureRect(IntRect(360, 380, SizeImg+10, SizeImg));

break;

}

case 2:

{

sprite.setTextureRect(IntRect(443, 384, SizeImg, SizeImg+5));

break;

}

case 3:

{

sprite.setTextureRect(IntRect(507, 380, SizeImg, SizeImg-10));

break;

}

case 4:

{

sprite.setTextureRect(IntRect(5, 376, SizeImg, SizeImg));

break;

}

}

break;

}

case north://изображение атаки при направлении вверх

{

switch (int(n))

{

case 0:

{

sprite.setTextureRect(IntRect(320, 0, SizeImg, SizeImg - 10));

break;

}

case 1:

{

sprite.setTextureRect(IntRect(382, 0, SizeImg - 10, SizeImg + 10));

break;

}

case 2:

{

sprite.setTextureRect(IntRect(446, 0, SizeImg + 10, SizeImg - 10));

break;

}

case 3:

{

sprite.setTextureRect(IntRect(505, 0, SizeImg - 10, SizeImg));

break;

}

case 4:

{

sprite.setTextureRect(IntRect(0, 0, SizeImg, SizeImg));

break;

}

}

break;

}

case south://изображение атаки при направлении вниз

{

switch (int(n))

{

case 0:

{

sprite.setTextureRect(IntRect(310, 252, SizeImg + 10, SizeImg));

break;

}

case 1:

{

sprite.setTextureRect(IntRect(380, 254, SizeImg, SizeImg - 10));

break;

}

case 2:

{

sprite.setTextureRect(IntRect(440, 246, SizeImg + 10, SizeImg));

break;

}

case 3:

{

sprite.setTextureRect(IntRect(508, 249, SizeImg - 10, SizeImg + 10));

break;

}

case 4:

{

sprite.setTextureRect(IntRect(0, 248, SizeImg, SizeImg));

break;

}

}

break;

}

case SouthEast://изображение атаки при направлении вниз-вправо

{

switch (int(n))

{

case 0:

{

sprite.setTextureRect(IntRect(311, 189, SizeImg + 10, SizeImg - 10));

break;

}

case 1:

{

sprite.setTextureRect(IntRect(375, 188, SizeImg, SizeImg + 10));

break;

}

case 2:

{

sprite.setTextureRect(IntRect(436, 190, SizeImg + 10, SizeImg));

break;

}

case 3:

{

sprite.setTextureRect(IntRect(505, 184, SizeImg + 20, SizeImg));

break;

}

case 4:

{

sprite.setTextureRect(IntRect(0, 185, SizeImg, SizeImg));

break;

}

}

break;

}

case SouthWest://изображение атаки при направлении вниз-влево

{

switch (int(n))

{

case 0:

{

sprite.setTextureRect(IntRect(331, 316, SizeImg - 10, SizeImg + 10));

break;

}

case 1:

{

sprite.setTextureRect(IntRect(383, 319, SizeImg, SizeImg + 10));

break;

}

case 2:

{

sprite.setTextureRect(IntRect(448, 317, SizeImg - 5, SizeImg + 10));

break;

}

case 3:

{

sprite.setTextureRect(IntRect(491, 315, SizeImg + 20, SizeImg));

break;

}

case 4:

{

sprite.setTextureRect(IntRect(0, 313, SizeImg, SizeImg));

break;

}

}

break;

}

case NorthEast://изображение атаки при направлении вверх-вправо

{

switch (int(n))

{

case 0:

{

sprite.setTextureRect(IntRect(317, 60, SizeImg + 15, SizeImg));

break;

}

case 1:

{

sprite.setTextureRect(IntRect(383, 62, SizeImg+10, SizeImg));

break;

}

case 2:

{

sprite.setTextureRect(IntRect(444, 58, SizeImg + 10, SizeImg + 10));

break;

}

case 3:

{

sprite.setTextureRect(IntRect(505, 63, SizeImg, SizeImg));

break;

}

case 4:

{

sprite.setTextureRect(IntRect(0, 58, SizeImg, SizeImg));

break;

}

}

break;

}

case NorthWest://изображение атаки при направлении вверх-влево

{

switch (int(n))

{

case 0:

{

sprite.setTextureRect(IntRect(306, 445, SizeImg + 10, SizeImg));

break;

}

case 1:

{

sprite.setTextureRect(IntRect(369, 445, SizeImg + 10, SizeImg));

break;

}

case 2:

{

sprite.setTextureRect(IntRect(429, 448, SizeImg + 10, SizeImg));

break;

}

case 3:

{

sprite.setTextureRect(IntRect(505, 446, SizeImg + 20, SizeImg));

break;

}

case 4:

{

sprite.setTextureRect(IntRect(5, 442, SizeImg, SizeImg));

break;

}

}

break;

}

}

}

if (hp <= 0)

{

life = false;

sprite.setTextureRect(IntRect(700, 188, SizeImg, SizeImg));

}

sprite.setPosition(PosX, PosY);

}

}

-----------------------------------------------------------------------------------------

//Magma.cpp

#include "Magma.h"

Magma::Magma(float x, float y)

{

PrevState = east;//направляющее положение

state = stay; // начальное положение

hp = 300;

PosX = x;

PosY = y;

speed = 0;

type = magma;

CurrentFrame = 0;

life = true;

damage = -1;

texture.loadFromFile("images/map.png");

sprite.setTexture(texture);

sprite.setPosition(PosX, PosY);

sprite.setTextureRect(IntRect(175, 225, SizeImgMap, SizeImgMap));

}

bool Magma::RadiusDamage(float x,float y)

{

return true;

}

void Magma::SetHp(int h, Gtype t)

{

hp = hp;

}

void Magma::control()

{

state = stay;

}

void Magma::update(float time,float n1)

{

if (life)

{

control();

sprite.setTextureRect(IntRect(175, 225, SizeImgMap, SizeImgMap));

}

sprite.setPosition(PosX, PosY);

}

-----------------------------------------------------------------------------------------

//MapCreator.cpp

#include "MapCreator.h"

char TileMap[HeightMap][HeightMap];

void RandomMapGenerate() //функция для создания карты

{

int RandomNumber;

for (int i = 0; i < HeightMap; i++)

for (int j = 0; j < HeightMap; j++)

{

RandomNumber = rand() % 100 ;

if ((RandomNumber >= 0) && (RandomNumber < 50))

{

TileMap[i][j] = '0';

}

if ((RandomNumber >= 50) && (RandomNumber < 75))

{

TileMap[i][j] = '1';

}

if ((RandomNumber >= 75) && (RandomNumber < 98))

{

TileMap[i][j] = '2';

}

if ((RandomNumber >= 98) && (RandomNumber < 100))

{

TileMap[i][j] = 'm';

}

}

}

void UpdateMap(int n)

{

int RandomNumber;

char AddTileMap[HeightMap][HeightMap];

// сдвиг вправо на 1 ячейку

if (n == 1)

{

for (int i = 0;i < HeightMap;i++)

for (int j = 1;j < HeightMap;j++)

{

AddTileMap[i][j] = TileMap[i][j-1];//копирование карты со сдвигом вправо на 1 ячейку

}

for(int i = 0;i < HeightMap;i++)//добавление левой линии карты

{

RandomNumber = rand() % 10000 ;

if ((RandomNumber >= 0) && (RandomNumber < 4984)) AddTileMap[i][0] = '0';

if ((RandomNumber >= 4983) && (RandomNumber < 4984)) AddTileMap[i][0] = 'h';

if ((RandomNumber >= 4984) && (RandomNumber < 4985)) AddTileMap[i][0] = 'd';

if ((RandomNumber >= 4985) && (RandomNumber < 4990)) AddTileMap[i][0] = 't';

if ((RandomNumber >= 4990) && (RandomNumber < 5000)) AddTileMap[i][0] = 'g';

if ((RandomNumber >= 5000) && (RandomNumber < 7500)) AddTileMap[i][0] = '1';

if ((RandomNumber >= 7500) && (RandomNumber < 9800)) AddTileMap[i][0] = '2';

if ((RandomNumber >= 9800) && (RandomNumber < 10000)) AddTileMap[i][0] = 'm';

}

}

// сдвиг влево на 1 ячейку

if (n == 2)

{

for (int i = 0;i < HeightMap;i++)

for (int j = 0;j < (HeightMap - 1);j++)

{

AddTileMap[i][j] = TileMap[i][j+1];//копирование карты со сдвигом влево на 1 ячейку

}

for(int i = 0;i < HeightMap;i++)//добавление правой линии карты

{

RandomNumber = rand() % 10000 ;

if ((RandomNumber >= 0) && (RandomNumber < 4984)) AddTileMap[i][HeightMap - 1] = '0';

if ((RandomNumber >= 4983) && (RandomNumber < 4984)) AddTileMap[i][HeightMap - 1] = 'h';

if ((RandomNumber >= 4984) && (RandomNumber < 4985)) AddTileMap[i][HeightMap - 1] = 'd';

if ((RandomNumber >= 4985) && (RandomNumber < 4990)) AddTileMap[i][HeightMap - 1] = 't';

if ((RandomNumber >= 4990) && (RandomNumber < 5000)) AddTileMap[i][HeightMap - 1] = 'g';

if ((RandomNumber >= 5000) && (RandomNumber < 7500)) AddTileMap[i][HeightMap - 1] = '1';

if ((RandomNumber >= 7500) && (RandomNumber < 9800)) AddTileMap[i][HeightMap - 1] = '2';

if ((RandomNumber >= 9800) && (RandomNumber < 10000)) AddTileMap[i][HeightMap - 1] = 'm';

}

}

// сдвиг вверх на 1 ячейку

if (n == 3)

{

for (int i = 0;i < (HeightMap - 1);i++)

for (int j = 0;j < HeightMap;j++)

{

AddTileMap[i][j] = TileMap[i+1][j];//копирование карты со сдвигом вверх на 1 ячейку

}

for(int j = 0;j < HeightMap;j++)//добавление нижней линии карты

{

RandomNumber = rand() % 10000 ;

if ((RandomNumber >= 0) && (RandomNumber < 4984)) AddTileMap[HeightMap - 1][j] = '0';

if ((RandomNumber >= 4983) && (RandomNumber < 4984)) AddTileMap[HeightMap - 1][j] = 'h';

if ((RandomNumber >= 4984) && (RandomNumber < 4985)) AddTileMap[HeightMap - 1][j] = 'd';

if ((RandomNumber >= 4985) && (RandomNumber < 4990)) AddTileMap[HeightMap - 1][j] = 't';

if ((RandomNumber >= 4990) && (RandomNumber < 5000)) AddTileMap[HeightMap - 1][j] = 'g';

if ((RandomNumber >= 5000) && (RandomNumber < 7500)) AddTileMap[HeightMap - 1][j] = '1';

if ((RandomNumber >= 7500) && (RandomNumber < 9800)) AddTileMap[HeightMap - 1][j] = '2';

if ((RandomNumber >= 9800) && (RandomNumber < 10000)) AddTileMap[HeightMap - 1][j] = 'm';

}

}

// сдвиг вниз на 1 ячейку

if (n == 4)

{

for (int i = 1;i < HeightMap;i++)

for (int j = 0;j < HeightMap;j++)

{

AddTileMap[i][j] = TileMap[i-1][j];//копирование карты со сдвигом вниз на 1 ячейку

}

for(int j = 0;j < HeightMap;j++)//добавление верхней линии карты

{

RandomNumber = rand() % 10000 ;

if ((RandomNumber >= 0) && (RandomNumber < 4984)) AddTileMap[0][j] = '0';

if ((RandomNumber >= 4983) && (RandomNumber < 4984)) AddTileMap[0][j] = 'h';

if ((RandomNumber >= 4984) && (RandomNumber < 4985)) AddTileMap[0][j] = 'd';

if ((RandomNumber >= 4985) && (RandomNumber < 4990)) AddTileMap[0][j] = 't';

if ((RandomNumber >= 4990) && (RandomNumber < 5000)) AddTileMap[0][j] = 'g';

if ((RandomNumber >= 5000) && (RandomNumber < 7500)) AddTileMap[0][j] = '1';

if ((RandomNumber >= 7500) && (RandomNumber < 9800)) AddTileMap[0][j] = '2';

if ((RandomNumber >= 9800) && (RandomNumber < 10000)) AddTileMap[0][j] = 'm';

}

}

for (int i = 0;i < HeightMap;i++)

for (int j = 0;j < HeightMap;j++)

{

TileMap[i][j] = AddTileMap[i][j];

}

}

-----------------------------------------------------------------------------------------

//TowerWithArcher.cpp

#include "TowerWithArcher.h"

TowerArch::TowerArch(float x, float y)

{

PrevState = east;//направляющее положение

state = stay; // начальное положение

hp = 300;

PosX = x;

PosY = y;

speed = 0;

type = tow;

CurrentFrame = 0;

life = true;

damage = 0;

N = 0;

texture.loadFromFile("images/orc\_town.png");

sprite.setTexture(texture);

sprite.setPosition(PosX, PosY);

sprite.setTextureRect(IntRect(484, 435, SizeImgTown, SizeImgTown));

}

bool TowerArch::RadiusDamage(float x,float y)

{

if ((x - PosX) > 0)

{

if ((x - PosX) > SizeImg) return true;

else

{

if ((y - PosY) > 0)

{

if ((y - PosY) > SizeImg) return true;

else return false;

}

else

{

if ((PosY - y) > SizeImg) return true;

else false;

}

}

}

else

{

if ((PosX - x) > SizeImg) return true;

else

{

if ((y - PosY) > 0)

{

if ((y - PosY) > SizeImg) return true;

else return false;

}

else

{

if ((PosY - y) > SizeImg) return true;

else false;

}

}

}

}

void TowerArch::SetHp(int h, Gtype t)

{

hp += h;

}

void TowerArch::control()

{

state = stay;

}

void TowerArch::update(float time,float n1)

{

if (life)

{

control();

sprite.setTextureRect(IntRect(484, 435, SizeImgTown, SizeImgTown));

if (hp <= 0)

{

life = false;

sprite.setTextureRect(IntRect(240, 432, SizeImgTown, SizeImgTown));

}

}

sprite.setPosition(PosX, PosY);

}

-----------------------------------------------------------------------------------------

//main.cpp

#include <SFML/Graphics.hpp>

#include <iostream>

#include "Hero.h"

#include "globals.h"

#include "MapCreator.h"

#include "grant.h"

#include <list>

#include <sstream>

#include "TowerWithArcher.h"

#include "arrow.h"

#include "HealthBonus.h"

#include "DamageBonus.h"

#include "Magma.h"

using namespace sf;

using namespace std;

extern char TileMap[HeightMap][HeightMap];

int main()

{

Hero hero;

Clock clock;

Clock clock2;

Clock gameTimeClock;//переменная игрового времени, будем здесь хранить время игры

int gameTime = 0;//объявили игровое время, инициализировали.

Font font;

font.loadFromFile("CyrilicOld.TTF");

Text TextHp("", font, 40);

TextHp.setColor(Color::Green);

Text TextScore("", font, 40);

TextScore.setColor(Color::Green);

Text TextLose("", font, 100);

TextLose.setColor(Color::Red);

Text TextLose1("", font, 102);

TextLose.setColor(Color::Black);

std::list<Entity\*> enemy;

std::list<Entity\*>::iterator ItEnemy;

std::list<Arrow\*> arrow;

std::list<Arrow\*>::iterator ItArrow;

Image MapImage;

MapImage.loadFromFile("images/map.png");

Texture map;

map.loadFromImage(MapImage);//заряжаем текстуру картинкой

Sprite SpriteMap;//создаём спрайт для карты

SpriteMap.setTexture(map);//заливаем текстуру спрайтом

sf::VideoMode desktop = sf::VideoMode::getDesktopMode();

sf::RenderWindow window(sf::VideoMode(size, size, desktop.bitsPerPixel), "The Last Knight");

RandomMapGenerate();

while (window.isOpen()) //Пока окно открыто

{

static float n = 0;

int x = -50, y = -50;

float dx1 = 0, dy1 = 0;

float MapSpeed = 10\*hero.GetSpeed();

static float x1 = 0, y1 = 0;

Gstate status = hero.GetState();

float time = clock.getElapsedTime().asMicroseconds();

float AttackTime = clock2.getElapsedTime().asMilliseconds();

if (hero.GetLife()) gameTime = gameTimeClock.getElapsedTime().asSeconds();

//игровое время в секундах идёт вперед, пока жив игрок.

clock.restart();

time = time / 800;

sf::Event event;

while (window.pollEvent(event))

{

if (event.type == sf::Event::Closed)

window.close();

}

if ((int)n == 0)

{

if ((hero.GetState() == attack) && (hero.GetLife()) && (AttackTime >= delay / 3))

{

clock2.restart();

n += 0.01\*time;

}

else

{

switch (status)// установление изменения карты в зависимости от движения

{

case north:

{

dx1 = 0;

dy1 = MapSpeed;

break;

}

case west:

{

dx1 = MapSpeed;

dy1 = 0;

break;

}

case east:

{

dx1 = -MapSpeed;

dy1 = 0;

break;

}

case south:

{

dx1 = 0;

dy1 = -MapSpeed;

break;

}

case NorthWest:

{

dx1 = MapSpeed;

dy1 = MapSpeed;

break;

}

case NorthEast:

{

dx1 = -MapSpeed;

dy1 = MapSpeed;

break;

}

case SouthWest:

{

dx1 = MapSpeed;

dy1 = -MapSpeed;

break;

}

case SouthEast:

{

dx1 = -MapSpeed;

dy1 = -MapSpeed;

break;

}

default:

{

dx1 = 0;

dy1 = 0;

break;

}

}

}

}

x1 += dx1;

y1 += dy1;

for (int i = 0; i < HeightMap; i++) // отображение карты

{

for (int j = 0; j < HeightMap; j++)

{

if (TileMap[i][j] == '0') SpriteMap.setTextureRect(IntRect(0, 25, SizeImgMap, SizeImgMap));

if (TileMap[i][j] == '1') SpriteMap.setTextureRect(IntRect(75, 50, SizeImgMap, SizeImgMap));

if (TileMap[i][j] == '2') SpriteMap.setTextureRect(IntRect(175, 25, SizeImgMap, SizeImgMap));

if (TileMap[i][j] == '3') SpriteMap.setTextureRect(IntRect(175, 225, SizeImgMap, SizeImgMap));

if (TileMap[i][j] == 'g')//Добавление орка в лист если он заспаунился

{

TileMap[i][j] = '0';//обнуление ячейки в которой появился орк, чтобы там больше орков не появлялось

SpriteMap.setTextureRect(IntRect(0, 25, SizeImgMap, SizeImgMap));

enemy.push\_back(new Grant(x, y));

}

if (TileMap[i][j] == 't')//Добавление орка в лист если он заспаунился

{

TileMap[i][j] = '0';//обнуление ячейки в которой появилась башня, чтобы там больше башни не появлялись

SpriteMap.setTextureRect(IntRect(0, 25, SizeImgMap, SizeImgMap));

enemy.push\_back(new TowerArch(x, y));

}

if (TileMap[i][j] == 'h')

{

TileMap[i][j] = '0';//обнуление ячейки в которой появился бонус, чтобы там больше бонусы не появлялись

SpriteMap.setTextureRect(IntRect(0, 25, SizeImgMap, SizeImgMap));

enemy.push\_back(new HealthBonus(x, y));

}

if (TileMap[i][j] == 'd')

{

TileMap[i][j] = '0';//обнуление ячейки в которой появился бонус, чтобы там больше бонусы не появлялись

SpriteMap.setTextureRect(IntRect(0, 25, SizeImgMap, SizeImgMap));

enemy.push\_back(new DamageBonus(x, y));

}

if (TileMap[i][j] == 'm')//Добавление орка в лист если он заспаунился

{

TileMap[i][j] = '3';// добавление одного объекта магмы на клетку в которой он появился

enemy.push\_back(new Magma(x, y));

}

SpriteMap.setPosition((int)(x + x1), (int)(y + y1));

window.draw(SpriteMap);

x += 25;

}

x = -50;

y += 25;

}

// проверка: сдвинулась ли карта на ячейку по какой либо координате

if (((int) x1) == SizeImgMap)

{

UpdateMap(1);

x1 = 0;

}

if (((int) x1) == -SizeImgMap)

{

UpdateMap(2);

x1 = 0;

}

if (((int) y1) == -SizeImgMap)

{

UpdateMap(3);

y1 = 0;

}

if (((int) y1) == SizeImgMap)

{

UpdateMap(4);

y1 = 0;

}

if (n > 0)//проверка атаки по врагу

{

n += 0.01\*time;

if ((int)n == 4 )

{

for (ItEnemy = enemy.begin();ItEnemy != enemy.end(); ItEnemy++)

{

if ((hero.RadiusDamage((\*ItEnemy)->GetPosX(), (\*ItEnemy)->GetPosY()) == true) && ((\*ItEnemy)->GetLife()))

{

if (hero.TimeOfAction())

{

(\*ItEnemy)->SetHp(2 \* hero.GetDamage(), hero.GetType());

}

else (\*ItEnemy)->SetHp(hero.GetDamage(),hero.GetType());

hero.SetScore(20);

}

}

}

}

//выстрел башни

if (hero.GetLife())

{

for (ItEnemy = enemy.begin();ItEnemy != enemy.end(); ItEnemy++)

{

if ((\*ItEnemy)->GetType() == tow)

{

if (((\*ItEnemy)->GetLife()) && (\*ItEnemy)->ShootDelay())

{

arrow.push\_back(new Arrow((\*ItEnemy)->GetPosX(), (\*ItEnemy)->GetPosY()));

}

}

}

}

//изменение позиции стрел, если герой сдвинулся

for (ItArrow = arrow.begin();ItArrow != arrow.end();ItArrow++)

{

(\*ItArrow)->ChangePos(dx1,dy1);

}

//изменение координат врагов относительно центра, если герой движется

for (ItEnemy = enemy.begin();ItEnemy != enemy.end(); ItEnemy++)

{

(\*ItEnemy)->ChangePos(dx1,dy1);

}

//оживление стрел

for (ItArrow = arrow.begin();ItArrow != arrow.end();ItArrow++)

{

(\*ItArrow)->update(time,0);

}

//обработка попадания стрел в игрока

for (ItArrow = arrow.begin();ItArrow != arrow.end();ItArrow++)

{

if (hero.GetRect().intersects((\*ItArrow)->GetRect()))

{

hero.SetHp((\*ItArrow)->GetDamage(),(\*ItArrow)->GetType());

(\*ItArrow)->SetLife(false);

}

}

//собирание бонусов и попадане в лаву

for (ItEnemy = enemy.begin();ItEnemy != enemy.end(); ItEnemy++)

{

if ((hero.GetRect().intersects((\*ItEnemy)->GetRect())) && ((\*ItEnemy)->GetType() == BonusHp))

{

hero.SetHp((\*ItEnemy)->GetDamage(),hero.GetType());

(\*ItEnemy)->SetLife(false);

}

if ((hero.GetRect().intersects((\*ItEnemy)->GetRect())) && ((\*ItEnemy)->GetType() == BonusDam))

{

hero.StartTime2();

(\*ItEnemy)->SetLife(false);

}

if ((hero.GetRect().intersects((\*ItEnemy)->GetRect())) && ((\*ItEnemy)->GetType() == magma))

{

hero.SetHp((\*ItEnemy)->GetDamage(),(\*ItEnemy)->GetType());

}

}

//оживление врагов

for (ItEnemy = enemy.begin();ItEnemy != enemy.end(); ItEnemy++)

{

if (((\*ItEnemy)->GetState() == attack) && ((\*ItEnemy)->GetLife()) && (\*ItEnemy)->ShootDelay() && (hero.GetLife())) // начало атаки

{

(\*ItEnemy)->SetN(time);

}

if ((\*ItEnemy)->GetN() > 0)// если атака начата

{

if (((int)((\*ItEnemy)->GetN()) == 4 ) && ((\*ItEnemy)->RadiusDamage(0,0))) hero.SetHp((\*ItEnemy)->GetDamage(),(\*ItEnemy)->GetType());

(\*ItEnemy)->SetN(time);

}

(\*ItEnemy)->update(time, (\*ItEnemy)->GetN());

}

//удаление врагов, если они вышли за пределы карты, и при этом мертвы, а также удаление бонусов если они подобраны

for (ItEnemy = enemy.begin();ItEnemy != enemy.end();)

{

if ((((\*ItEnemy)->GetPosX() < -50) || ((\*ItEnemy)->GetPosY() < -50) || ((\*ItEnemy)->GetPosX() > 850) || ((\*ItEnemy)->GetPosY() > 850)) && ((\*ItEnemy)->GetLife() == false))

{

ItEnemy = enemy.erase(ItEnemy);

}

else if ((((\*ItEnemy)->GetType() == BonusHp) || ((\*ItEnemy)->GetType() == BonusDam)) && ((\*ItEnemy)->GetLife() == false))

{

ItEnemy = enemy.erase(ItEnemy);

}

else ItEnemy++;

}

//удаление стрел, если они попали в героя или улетели за пределы карты

for (ItArrow = arrow.begin();ItArrow != arrow.end();)

{

if (((\*ItArrow)->GetPosX() < -50) || ((\*ItArrow)->GetPosY() < -50) || ((\*ItArrow)->GetPosX() > 850) || ((\*ItArrow)->GetPosY() > 850) || ((\*ItArrow)->GetLife() == false))

{

ItArrow = arrow.erase(ItArrow);

}

else ItArrow++;

}

//вывод на экран врагов

for (ItEnemy = enemy.begin();ItEnemy != enemy.end(); ItEnemy++)

{

window.draw((\*ItEnemy)->GetImage());

}

//вывод на экран пуль

for (ItArrow = arrow.begin();ItArrow != arrow.end();ItArrow++)

{

window.draw((\*ItArrow)->GetImage());

}

hero.SetStay();

//оживление героя

hero.update(time, n);

window.draw(hero.GetImage());

if (n > 4) n = 0;

//вывод очков и жизней на экран

std::ostringstream playerHPString;

std::ostringstream playerScoreString;

playerHPString << hero.GetHp();

playerScoreString << hero.GetScore();

TextHp.setString("Здоровье: " + playerHPString.str());

TextHp.setPosition(500, 0);

TextScore.setString("Очки: " + playerScoreString.str());

TextScore.setPosition(10, 0);

if (hero.GetLife() == false)

{

TextLose1.setString("ПОТРАЧЕНО");

TextLose1.setPosition(99, 349);

window.draw(TextLose1);

TextLose.setString("ПОТРАЧЕНО");

TextLose.setPosition(100, 350);

window.draw(TextLose);

}

window.draw(TextHp);

window.draw(TextScore);

window.display(); //Показываем объект на экране

window.clear(); //Очищаем экран

}

for (ItEnemy = enemy.begin(); ItEnemy != enemy.end();)

{

ItEnemy = enemy.erase(ItEnemy);

}

for (ItArrow = arrow.begin(); ItArrow != arrow.end();)

{

ItArrow = arrow.erase(ItArrow);

}

return 0;

}