**Отчет по командной работе. Горвиков Даниил.  
Пояснение сделанных коммитов**

Класс монстров

|  |  |  |  |
| --- | --- | --- | --- |
| Дата/Время | commit | комментарий | Цель создания |
| 2018-12-24 01:10:21 | f5110880bdb56c809bcec58e05434bfc1bd4602a | Jumper | Первая версия создания монстров |
| 2018-12-25 00:22:24 | 84edb0ablae382fbb2a096789b62b4d51d962194  0c225fd90b12cf5c3b4163ae8072c4050e4f74bf  73c47bb5b488dd983f159c247dd2599ddb6a409 | enemy    enemy 2    V4 | Учел некоторые недостатки при пробном запуске игры и добавил функцию void |
| 2018-12-25 01:36:37 | f29bb0fe8b5dc650f0c48a75b3e7d94ad4ba55bd   cd8555d25e12cb3cfd02820566ec80d991bb256d   88ee919c7cf8a041c38bd59d231744150478a564    cd8555d25e12cb3cfd02820566ec80d991bb256d | enemyV502    enemyV501    enemyV50     enemyV5 | Окончательная версия мостров с учетом всех ошибок |

**Danya.cpp**  
 // обработка врагов е1, е2, е3

if (e1.enemyLive){

if (e1.enemysprite.getPosition().x > 1280-48)

e1.enemyMove = true;

if (e1.enemysprite.getPosition().x < 0)

e1.enemyMove = false;

if (!e1.enemyMove){

e1.enemysprite.move(0.2 \* time, 0);

changeEnemyPosition(e1.enemysprite.getPosition().x, e1.enemysprite.getPosition().y);

}

if (e1.enemyMove){

e1.enemysprite.move(-0.2 \* time, 0);

changeEnemyPosition(e1.enemysprite.getPosition().x, e1.enemysprite.getPosition().y);

}

fightWithAShadow(48,64);

if (niceShoot(48,64) && generatebullet){

e1.lives --;

generatebullet = false;

pow.bulletLive = false;

}

if (e1.lives == 0)

e1.enemyLive = false;

}

if (e2.enemyLive){

if (e2.enemysprite.getPosition().x > 1280-64)

e2.enemyMove = true;

if (e2.enemysprite.getPosition().x < 0)

e2.enemyMove = false;

if (!e2.enemyMove){

e2.enemysprite.move(0.6 \* time, 0);

changeEnemyPosition(e2.enemysprite.getPosition().x, e2.enemysprite.getPosition().y);

}

if (e2.enemyMove){

e2.enemysprite.move(-0.6 \* time, 0);

changeEnemyPosition(e2.enemysprite.getPosition().x, e2.enemysprite.getPosition().y);

}

fightWithAShadow(64,64);

if (niceShoot(64,64) && generatebullet){

e2.lives --;

generatebullet = false;

pow.bulletLive = false;

}

if (e2.lives == 0)

e2.enemyLive = false;

}

if (e3.enemyLive){

if (generateEnemyBullet){

enemystrikes.bulletLive = true;

}

if (enemystrikes.bulletLive){

enemystrikes.bulletsprite.move(0.7 \* enemystrikes.dirBullet \* time, 0);

changeEnemyBulletPosition(enemystrikes.bulletsprite.getPosition().x, enemystrikes.bulletsprite.getPosition().y);

if (enemystrikes.bulletsprite.getPosition().x > 1280 || enemystrikes.bulletsprite.getPosition().x < 0){

enemystrikes.bulletLive = false;

}

}

if (niceShoot(128,128) && generatebullet){

e3.lives --;

generatebullet = false;

pow.bulletLive = false;

changeEnemyPosition(e3.enemysprite.getPosition().x, e3.enemysprite.getPosition().y);

}

if (e3.lives == 0)

e3.enemyLive = false;

if (!enemystrikes.bulletLive){

generateEnemyBullet = true;

enemystrikes.bulletsprite.setPosition(e3.enemysprite.getPosition().x + 128, e3.enemysprite.getPosition().y + 64);

changeEnemyBulletPosition(enemystrikes.bulletsprite.getPosition().x, enemystrikes.bulletsprite.getPosition().y);

}

if (comradYouHaveAHole())

enemystrikes.bulletLive = false;

} else enemystrikes.bulletLive = false;

**Danya1.h**class enemy {

public:

bool enemyMove;

bool enemyLive;

int lives;

Image image;

Texture texture;

Sprite enemysprite;

enemy(float X, float Y){

image.loadFromFile("bones.png");

texture.loadFromImage(image);

enemysprite.setTexture(texture);

enemysprite.setTextureRect(IntRect(48, 128, 48, 64));

enemysprite.setPosition(X, Y);

}

};

class enemy2 {

public:

bool enemyMove;

bool enemyLive;

int lives;

Image image;

Texture texture;

Sprite enemysprite;

enemy2(float X, float Y){

image.loadFromFile("ghost.png");

texture.loadFromImage(image);

enemysprite.setTexture(texture);

enemysprite.setTextureRect(IntRect(64, 48, 64, 64));

enemysprite.setPosition(X, Y);

}

};

class enemy3 {

public:

bool enemyLive;

int lives;

Image image;

Texture texture;

Sprite enemysprite;

enemy3(float X, float Y){

image.loadFromFile("ghost.png");

texture.loadFromImage(image);

enemysprite.setTexture(texture);

enemysprite.setTextureRect(IntRect(64\*16, 16, 128, 128));

enemysprite.setPosition(X, Y);

}

};

**V4.cpp**  
#include "core.hpp"  
#include <iostream>  
#include <stdio.h>  
#include <fstream>  
#include <string>  
  
using namespace std;  
  
class Hero {  
public:  
 int x;  
 int y;  
 float lives;  
};  
  
struct Platform {  
 int x;  
 int y;  
};  
  
struct Enemy{  
 int x;  
 int y;  
};  
  
struct Bullet{  
 int x;  
 int y;  
};  
  
int score = 0;  
int newY = 0;  
string name = "JohnyBigBazooka";  
  
Hero mainHero; // Ñîçäàíèå ãëàâíîãî ãåðîÿ  
Platform platform;  
Platform movingPlatform;  
  
Platform arrayOfPlatforms[9];  
  
Enemy ghost;  
Bullet enemyBullet;  
Bullet heroBullet;  
  
int quantityOfPlatforms = 1;  
int control = 0;  
  
int convertYCoordinate(int y){  
 return(720 - y);  
}  
  
  
  
void startGame(int x, int y){  
   
 mainHero.x = x;  
 mainHero.y = convertYCoordinate(y);  
 mainHero.lives = 3;  
   
}  
  
  
  
void getPlatformCoordinates(int x1, int y1, int x2, int y2){  
 platform.x = x1;  
 platform.y = convertYCoordinate(y1);  
 movingPlatform.x = x2;  
 movingPlatform.y = convertYCoordinate(y2);  
 arrayOfPlatforms[quantityOfPlatforms - 1] = platform;  
 quantityOfPlatforms += 1;  
}  
  
void changePositionOfMovingPlatform(int x, int y){  
 movingPlatform.x = x;  
 movingPlatform.y = convertYCoordinate(y);  
}  
  
void changeHeroPosition(int x, int y){  
   
 mainHero.x = x;  
 mainHero.y = convertYCoordinate(y);  
   
}  
  
int gameOver(bool enemyAlive){  
 if (mainHero.lives <= 0) {  
 return 2;  
 }  
 if (mainHero.y - 32 <= 32){  
   
 mainHero.lives -= 0.5;  
   
 if (quantityOfPlatforms == 2 && enemyAlive && (mainHero.lives == 2 || mainHero.lives == 1)){  
 quantityOfPlatforms -= 1;  
 }  
 if (quantityOfPlatforms > 2 && enemyAlive && (mainHero.lives == 2 || mainHero.lives == 1)){  
 quantityOfPlatforms -= 1;  
 }  
 if (quantityOfPlatforms >= 3 && (mainHero.lives == 2 || mainHero.lives == 1)){  
 quantityOfPlatforms -= 2;  
 }  
   
   
 cout << mainHero.lives << "\n";  
 return 1;  
 }  
 return 0;  
}  
  
int onGround(){// âîçâðàùàåò 1 åñëè ñòîèò íà êîëîííå, 2 åñëè ñòîèò íà äâèãóþùåéñÿ ïëàòôîðìå, 0 åñëè íå ñòîèò  
   
 for (int i = 0; i < quantityOfPlatforms; i++){  
 if ((mainHero.x >= arrayOfPlatforms[i].x && mainHero.x <= arrayOfPlatforms[i].x + 256) && (mainHero.y - 128 <= arrayOfPlatforms[i].y))  
 return 1;  
 }  
   
   
   
 if ((mainHero.x >= movingPlatform.x && mainHero.x <= movingPlatform.x + 256) && (mainHero.y - 128 <= movingPlatform.y)){  
 return 2;  
 }  
   
 return 0;  
   
}  
  
void changeEnemyPosition(int x, int y){  
 ghost.x = x;  
 ghost.y = convertYCoordinate(y);  
}  
  
void changeEnemyBulletPosition(int x, int y){  
 enemyBullet.x = x;  
 enemyBullet.y = convertYCoordinate(y);  
}  
  
void changeHeroBulletPostion(int x, int y){  
 heroBullet.x = x;  
 heroBullet.y = convertYCoordinate(y);  
}  
  
  
int crash() { // ñòîëêíîâåíèå ñ äâèãàþùèéñÿ ïëàòôîðìîé  
 if ((mainHero.y - 128 >= platform.y) || (mainHero.y <= platform.y - 64)){  
 if (mainHero.x + 64 >= movingPlatform.x + 256+64 && mainHero.x + 64 <= movingPlatform.x + 256+64+5)  
 return 1;  
 else  
 if (mainHero.x >= movingPlatform.x - 69 && mainHero.x <= movingPlatform.x - 64 )  
 return 2;  
 }  
 return 0;  
}  
  
void fightWithAShadow(int width, int height){  
 if ((mainHero.x + 15 <= ghost.x + width && mainHero.x + 49 >= ghost.x) && (mainHero.y - 118 <= ghost.y) && (mainHero.y - 15 >= ghost.y - height)) mainHero.lives -= 3;  
}  
  
bool comradYouHaveAHole(){  
 if (mainHero.x + 64 >= enemyBullet.x && mainHero.x <= enemyBullet.x + 15 && enemyBullet.y <= mainHero.y && enemyBullet.y >= mainHero.y - 128) {  
 mainHero.lives -= 1;  
 enemyBullet.x = -100;  
 enemyBullet.y = -100;  
 return true;  
 }  
 return false;  
}  
  
bool niceShoot(int width, int height){  
 if ((ghost.x + 15 <= heroBullet.x + width) && (heroBullet.x <= ghost.x + width - 15) && (heroBullet.y - 14 >= ghost.y - height) && (heroBullet.y <= ghost.y)){  
 ghost.x = -100;  
 ghost.y = -100;  
 score += 1;  
 return true;  
 }  
 return false;  
}  
  
bool stopPlatform(){  
 if (onGround() == 2){  
 if (movingPlatform.x + 240 >= platform.x && movingPlatform.x <= platform.x + 240) return true;  
 score += 1;  
 }  
 return false;  
}  
  
int heroLives(){  
 return mainHero.lives;  
}  
  
int objectGenerator(){  
 int a = rand() % (quantityOfPlatforms + 4) +1;  
 if (a == 1){  
 return 1;  
 }  
 if (a > 7){  
 return 4;  
 }  
 if (a > 3 && a < 7){  
 return 3;  
 }  
 if (a >= 2 && a < 4){  
 return 2;  
 }  
 return 0;  
}  
  
int stop(){ // 1 - right 2 - left, 3 - top 0 - ok  
 if (mainHero.x <= 0) return 2;  
 if (mainHero.x + 64 >= 1280) return 1;  
 if (mainHero.y >= 720) return 3;  
   
 for (int i = 0; i < quantityOfPlatforms; i++){  
 if ((arrayOfPlatforms[i].y <= mainHero.y) && (arrayOfPlatforms[i].y - 64 >= mainHero.y - 128)){  
 if (arrayOfPlatforms[i].x <= mainHero.x + 64){  
 return 1;  
 }  
 if (arrayOfPlatforms[i].x + 256 <= mainHero.x){  
 return 2;  
 }  
 }  
 }  
 return 0;  
}  
  
int whatIsScore(){  
 return score;  
}  
  
int finish(){  
 if (mainHero.lives == 3) score += 10;  
 if (mainHero.lives == 2) score += 5;  
 if (mainHero.lives == 1) score += 3;  
 score += 10; // çà ôèíèø  
   
 ifstream file;  
 file.open("score.txt", ios::in);  
 string line;  
 string arrayOfLines[100];  
 int i = 0;  
 bool yes = false;  
 while (getline(file, line)) {  
   
 if (yes){  
 arrayOfLines[i] = score;  
 yes = false;  
 } else {  
 if (line == (name + "\n")){  
 yes = true;  
 arrayOfLines[i] = line;  
 } else {  
 arrayOfLines[i] = true;  
 }  
 }  
 i++;  
 }  
 file.close();  
   
 remove("score.txt");  
   
 ofstream newFile;  
 newFile.open("score.txt", ios::out);  
   
 for (int c = 0; c < i; i++){  
 newFile << arrayOfLines[c];  
 }  
   
 newFile.close();  
   
 return score;  
}  
  
bool endOfJump(){  
 if (onGround() != 0){  
 newY = mainHero.y;  
 return false;  
 } else {  
 if (mainHero.y - newY >= 200){  
 return true;  
 }  
 }  
 return false;  
}  
  
void whatIsQuantityOfPlatform(int count){  
 quantityOfPlatforms = count;  
}  
  
  
**V40.h**

|  |  |
| --- | --- |
| class enemy { |  |
|  | public: |
|  | bool enemyMove; |
|  | bool enemyLive; |
|  | int lives; |
|  | Image image; |
|  | Texture texture; |
|  | Sprite enemysprite; |
|  |  |
|  | enemy(float X, float Y){ |
|  | image.loadFromFile("bones.png"); |
|  | texture.loadFromImage(image); |
|  | enemysprite.setTexture(texture); |
|  | enemysprite.setTextureRect(IntRect(48, 128, 48, 64)); |
|  | enemysprite.setPosition(X, Y); |
|  | } |
|  | }; |
|  |  |
|  | class enemy2 { |
|  | public: |
|  | bool enemyMove; |
|  | bool enemyLive; |
|  | int lives; |
|  | Image image; |
|  | Texture texture; |
|  | Sprite enemysprite; |
|  |  |
|  | enemy2(float X, float Y){ |
|  | image.loadFromFile("ghost.png"); |
|  | texture.loadFromImage(image); |
|  | enemysprite.setTexture(texture); |
|  | enemysprite.setTextureRect(IntRect(64, 48, 64, 64)); |
|  | enemysprite.setPosition(X, Y); |
|  | } |
|  | }; |
|  |  |
|  | class enemy3 { |
|  | public: |
|  | bool enemyLive; |
|  | int lives; |
|  | Image image; |
|  | Texture texture; |
|  | Sprite enemysprite; |
|  |  |
|  | enemy3(float X, float Y){ |
|  | image.loadFromFile("ghost.png"); |
|  | texture.loadFromImage(image); |
|  | enemysprite.setTexture(texture); |
|  | enemysprite.setTextureRect(IntRect(64\*16, 16, 128, 128)); |
|  | enemysprite.setPosition(X, Y); |
|  | } |
|  | }; |

**V401.h**

|  |  |
| --- | --- |
| **#ifndef core\_hpp** |  |
|  | **#define core\_hpp** |
|  |  |
|  | **#include <stdio.h>** |
|  |  |
|  | **void startGame(int x, int y);** |
|  | **void getPlatformCoordinates(int x1, int y1, int x2, int y2);** |
|  | **void changePositionOfMovingPlatform(int x, int y);** |
|  | **void changeHeroPosition(int x, int y);** |
|  | **int gameOver(bool enemyAlive);** |
|  | **int onGround();** |
|  | **void changeEnemyPosition(int x, int y);** |
|  | **void changeEnemyBulletPosition(int x, int y);** |
|  | **void changeHeroBulletPostion(int x, int y);** |
|  | **int crash();** |
|  | **void fightWithAShadow(int width, int height);** |
|  | **bool comradYouHaveAHole();** |
|  | **bool niceShoot(int width, int height);.** |
|  | **bool stopPlatform();** |
|  | **int heroLives();** |
|  | **int objectGenerator();** |
|  | **int stop();** |
|  | **int whatIsScore();** |
|  | **int finish();** |
|  | **bool endOfJump();** |
|  | **void whatIsQuantityOfPlatform(int count);** |
|  |  |
|  | **#endif /\* core\_hpp \*/** |

**V5.cpp**bool enemyLive = false; //Жив монстр

e1.enemyMove = false; //Начальные параметры монстров

e1.enemyLive = false;

e2.enemyMove = false;

e2.enemyLive = false;

e3.enemyLive = false;

e1.lives = 2; //Жизни монстров

e2.lives = 4;

e3.lives = 4;

float CurrentFrame = 0; //хранит текущий кадр для анимации

Clock clock; //Время

int p = 0;

bool GameNotOver = true; //Чтобы выполнялся цикл

while (GameNotOver){

a = rand() % 2 +1; //Рандомная переменная для генерации платформ и монстров %2 - 2 числа

on\_ground = onGround(); //Вызов функции из core.cpp

if (gameOver(enemyLive) == 2) {

cout << finish();

NumOfWindow = 4;

GameNotOver = false;

}

if (countOfPlatform == 8){

NumOfWindow = 3;

GameNotOver = false;

}

if (gameOver(enemyLive) == 1) {

if (!platformCantMove)

// обработка врагов е1, е2, е3

if (e1.enemyLive){

if (e1.enemysprite.getPosition().x > 1280-48)

e1.enemyMove = true;

if (e1.enemysprite.getPosition().x < 0)

e1.enemyMove = false;

if (!e1.enemyMove){

e1.enemysprite.move(0.2 \* time, 0);

changeEnemyPosition(e1.enemysprite.getPosition().x, e1.enemysprite.getPosition().y);

}

if (e1.enemyMove){

e1.enemysprite.move(-0.2 \* time, 0);

changeEnemyPosition(e1.enemysprite.getPosition().x, e1.enemysprite.getPosition().y);

}

fightWithAShadow(48,64);

if (niceShoot(48,64) && generatebullet){

e1.lives --;

generatebullet = false;

pow.bulletLive = false;

}

if (e1.lives == 0)

e1.enemyLive = false;

}

if (e2.enemyLive){

if (e2.enemysprite.getPosition().x > 1280-64)

e2.enemyMove = true;

if (e2.enemysprite.getPosition().x < 0)

e2.enemyMove = false;

if (!e2.enemyMove){

e2.enemysprite.setScale(-1, 1);

e2.enemysprite.move(0.6 \* time, 0);

changeEnemyPosition(e2.enemysprite.getPosition().x, e2.enemysprite.getPosition().y);

}

if (e2.enemyMove){

e2.enemysprite.setScale(1, 1);

e2.enemysprite.move(-0.6 \* time, 0);

changeEnemyPosition(e2.enemysprite.getPosition().x, e2.enemysprite.getPosition().y);

}

fightWithAShadow(64,64);

if (niceShoot(64,64) && generatebullet){

e2.lives --;

generatebullet = false;

pow.bulletLive = false;

}

if (e2.lives == 0)

e2.enemyLive = false;

}

if (e3.enemyLive){

if (generateEnemyBullet){

enemystrikes.bulletLive = true;

}

if (enemystrikes.bulletLive){

if (enemystrikes.dirBullet == 1)

enemystrikes.bulletsprite.setScale(1, 1);

else

enemystrikes.bulletsprite.setScale(-1, 1);

enemystrikes.bulletsprite.move(0.6 \* enemystrikes.dirBullet \* time, 0);

changeEnemyBulletPosition(enemystrikes.bulletsprite.getPosition().x, enemystrikes.bulletsprite.getPosition().y);

if (enemystrikes.bulletsprite.getPosition().x > 1280 || enemystrikes.bulletsprite.getPosition().x < 0){

enemystrikes.bulletLive = false;

}

}

if (niceShoot(128,128) && generatebullet){

e3.lives --;

generatebullet = false;

pow.bulletLive = false;

changeEnemyPosition(e3.enemysprite.getPosition().x, e3.enemysprite.getPosition().y);

}

if (e3.lives == 0)

e3.enemyLive = false;

if (!enemystrikes.bulletLive){

generateEnemyBullet = true;

enemystrikes.bulletsprite.setPosition(e3.enemysprite.getPosition().x + 128, e3.enemysprite.getPosition().y + 64);

changeEnemyBulletPosition(enemystrikes.bulletsprite.getPosition().x, enemystrikes.bulletsprite.getPosition().y);

}

if (comradYouHaveAHole())

enemystrikes.bulletLive = false;

} else enemystrikes.bulletLive = false;

**V50.cpp**

struct Enemy{

int x;

int y;

};

if (mainHero.lives == 3) score += 10;

if (mainHero.lives == 2) score += 5;

if (mainHero.lives == 1) score += 3;

score += 10; // за финиш

**V501** .**cpp**

enemy(float X, float Y)

{

image.loadFromFile("bones.png");

texture.loadFromImage(image);

enemysprite.setTexture(texture);

enemysprite.setTextureRect(IntRect(48, 128, 48, 64));

enemysprite.setPosition(X, Y);

}

};

class enemy2 {

public:

bool enemyMove;

bool enemyLive;

int lives;

Image image;

Texture texture;

Sprite enemysprite;

enemy2(float X, float Y){

image.loadFromFile("ghost.png");

texture.loadFromImage(image);

enemysprite.setTexture(texture);

enemysprite.setTextureRect(IntRect(64, 48, 64, 64));

enemysprite.setPosition(X, Y);

}

};

class enemy3 {

public:

bool enemyLive;

int lives;

Image image;

Texture texture;

Sprite enemysprite;

enemy3(float X, float Y){

image.loadFromFile("ghost.png");

texture.loadFromImage(image);

enemysprite.setTexture(texture);

enemysprite.setTextureRect(IntRect(64\*16, 16, 128, 128));

enemysprite.setPosition(X, Y);

}

};

**V502.h**

void changeEnemyPosition(int x, int y); // Вызывать каждый раз при движении врага. (Принимает на борт координаты врага)

void changeEnemyBulletPosition(int x, int y); // При движении вражеской пули. (Координаты пули)

void fightWithAShadow(int width, int height); // Вызывать всегда (передавть ширину монстра, который сейчас на экране)

bool niceShoot(int width, int height); //Вызывать всегда. Возвращает true при попадании во врага. Принимает ширину врага.