Министерство науки и высшего образования 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»

Индивидуальный отчет

**Автор работы:**

студент группы 21312

Кобылин Дмитрий

**Научный руководитель:**

канд. физ.-мат. наук

А.В. Бульба

Петрозаводск 2023

# **Цель работы.**

Используя возможности языка программирования С++ (IDE Qt Creator (Qt 5.4.2)) и библиотеки (Simple and Fast Multimedia Library) написать простую 2D игру.

В качестве примера был выбрана игра Ведьмак 4 в стиле средневековой РПГ с видом сверху.

**Личные задачи, которые были решены:**

1. Получение врагами урона.
2. Создание второго вида оружия ближнего боя.
3. Решение логических багов (стрельба при ходьбе по диагонали, обработка событий на разных уровнях).
4. Добавление босса на второй уровень игры.

**История GITK:**



Всего было произведено 5 выгрузок, 3 из которых имеют массивные изменения.

**Решение каждой задачи:**

1. **Получение врагами урона (добавление текстуры врага).**

**Была создана и загружена текстура противника.**

|  |  |
| --- | --- |
|  | Image easyEnemyImage;  easyEnemyImage.loadFromFile("images/enemy.png");  …  …  enemies.push\_back(new Enemy(easyEnemyImage, xr, yr, 96, 96, "EasyEnemy")); |

Получение врагами урона:

|  |  |
| --- | --- |
| Получение урона реализовано через сравнение координат каждого элемента из списка Bullets и Stones с координатами каждого элемента списка enemies.  При пересечении элементы из обоих контейнеров удаляются методом erase(). |  |

Нанесение врагами урона:

|  |  |
| --- | --- |
| Нанесение врагами урона реализовано через сравнение координат каждого элемента контейнера enemies с координатами объекта p класса player | Player p(heroImage, 100, 100, 96, 96, "Player1"); |

1. **Создание второго вида оружия.**

Создан класс Stone в котором прописана логика поведения элемента в игре (разбор класса в видео-отчете):

class Stone :public Entity{

public:

int direction;

Stone(Image &image, float X, float Y, int W, int H, std::string Name, int dir)

:Entity(image, X, Y, W, H, Name){

x = X;

y = Y;

x1 = X;

y1 = Y;

direction = dir;

speed = 0.8;

w = h = 16;

life = true;

}

void *update*(float time)

{

switch (direction)

{

case 0: dx = -speed; dy = 0; break;

case 1: dx = speed; dy = 0; break;

case 2: dx = 0; dy = -speed; break;

case 3: dx = 0; dy = speed; break;

case 4: dx = speed; dy = -speed; break;

case 5: dx = -speed/2; dy = -speed/2; break;

case 6: dx = speed/2; dy = -speed/2; break;

case 7: dx = -speed/2; dy = speed/2; break;

case 8: dx = speed/2; dy = speed/2; break;

}

if (life){

x += dx\*1.1;

y += dy\*1.1;

if (x <= 0) x = 20;

if (y <= 0) y = 20;

if (x >= 800) x = 780;

if (y >= 800) y = 780;

for (int i = y / 32; i < (y + h) / 32; i++)

for (int j = x / 32; j < (x + w) / 32; j++)

{

if ((TileMap[i][j] == '0'))

life = false;

if ((Map[i][j] == '0'))

life = false;

if ((abs(x-x1)>100)||(abs(y-y1)>100))

life = false;

}

sprite.setPosition(x + w / 2, y + h / 2);

}

}

};

Была создана и загружена текстура камня:

|  |  |
| --- | --- |
|  | Image StoneImage;  StoneImage.loadFromFile("images/stone.png");  …  …  Stones.push\_back(new Stone(StoneImage, p.x, p.y, 16, 16, "Stone", p.state)); |

Реализовано столкновение с врагами для нанесения им урона:

Для врага:

if (p.life == true){

for (it = Stones.begin(); it != Stones.end(); it++){

for (it1 = enemies.begin(); it1 != enemies.end(); it1++){

if ((\*it)->getRect().intersects((\*it1)->getRect()))

{

it1 = enemies.erase(it1);

it = enemies.erase(it);

}

}

}

}

Обновление элемента:

for (it = Stones.begin(); it != Stones.end(); it++)

{

(\*it)->update(time);

}

1. **Обработка стрельбы при ходьбе по диагонали и логика обработки** **событий на разных уровнях.**

Ходьбу по диагонали прописывал **Тимофей Шестаков**:

if (life) {

control();

switch (state

{

case right:{

dx = speed;

CurrentFrame += 0.005\*time;

if (CurrentFrame > 3) CurrentFrame -= 3;

sprite.setTextureRect(IntRect(96 \* int(CurrentFrame), 192, 96, 96));

break;

}

case left:{

dx = -speed;

CurrentFrame += 0.005\*time;

if (CurrentFrame > 3) CurrentFrame -= 3;

sprite.setTextureRect(IntRect(96 \* int(CurrentFrame), 96, 96, 96));

break;

}

case up:{

dy = -speed;

CurrentFrame += 0.005\*time;

if (CurrentFrame > 3) CurrentFrame -= 3;

sprite.setTextureRect(IntRect(96 \* int(CurrentFrame), 288, 96, 96));

break;

}

case down:{

dy = speed;

CurrentFrame += 0.005\*time;

if (CurrentFrame > 3) CurrentFrame -= 3;

sprite.setTextureRect(IntRect(96 \* int(CurrentFrame), 0, 96, 96));

break;

}

case upright:{

dx = speed;

dy = -speed;

CurrentFrame += 0.005\*time;

if (CurrentFrame > 3) CurrentFrame -= 3;

sprite.setTextureRect(IntRect(96 \* int(CurrentFrame), 288, 96, 96));

break;

}

case upleft:{

dx = -speed;

dy = -speed;

CurrentFrame += 0.005\*time;

if (CurrentFrame > 3) CurrentFrame -= 3;

sprite.setTextureRect(IntRect(96 \* int(CurrentFrame), 288, 96, 96));

break;

}

case downright:{

dy = speed;

dx = speed;

CurrentFrame += 0.005\*time;

if (CurrentFrame > 3) CurrentFrame -= 3;

sprite.setTextureRect(IntRect(96 \* int(CurrentFrame), 0, 96, 96));

break;

}

case downleft:{

dy = speed;

dx = -speed;

CurrentFrame += 0.005\*time;

if (CurrentFrame > 3) CurrentFrame -= 3;

sprite.setTextureRect(IntRect(96 \* int(CurrentFrame), 0, 96, 96));

break;

}

case stay:{

dy = speed;

dx = speed;

break;

}

}

В методах Update классов Bullet и Stone необходимо было добавить обработку стрельбы во время диагонального перемещения:

switch (direction)

{

case 0: dx = -speed; dy = 0; break;

case 1: dx = speed; dy = 0; break;

case 2: dx = 0; dy = -speed; break;

case 3: dx = 0; dy = speed; break;

case 4: dx = speed; dy = -speed; break;

case 5: dx = -speed/2; dy = -speed/2; break;

case 6: dx = speed/2; dy = -speed/2; break;

case 7: dx = -speed/2; dy = speed/2; break;

case 8: dx = speed/2; dy = speed/2; break;

}

if (life){

x += dx\*time;

y += dy\*time;

…

}

В данном случае была добавлена обработка направления при значениях переменной direction = (5:8)

При этом переменным dx и dy присваивается переменная speed/2 т.к. в плоскости координат элементы должны передвигаться по диагонали.

Также были добавлены обработки текущего уровня в событиях, связанных с противниками (иначе появлялся баг, заключающийся в появлении монстров на уровне, не предназначенном для них: мобы спавнились на втором уровне)

if (Lvl == 1){

for (it = enemies.begin(); it != enemies.end(); it++)

{

if ((\*it)->life)

window.draw((\*it)->sprite);

}

}

\*Отрисовка монстров с учетом уровня.

**4. Добавление босса на второй уровень игры.**

Был создан класс Boss

class Boss :public Entity{

public:

int direction;

Boss(Image &image, float X, float Y, int W, int H, std::string Name) :Entity(image, X,

Y, W, H, Name){

if (name == "hardBoss"){

sprite.setTextureRect(IntRect(0, 0, w, h));

direction = rand() % (3);

speed = 0.15;

dx = speed;

}

}

void checkCollisionWithMap(float Dx, float Dy)

{

for (int i = y / 32; i < (y + h) / 32; i++)

for (int j = x / 32; j<(x + w) / 32; j++)

{

if (Map[i][j] == '0')

{

if (Dy > 0) {

y = i \* 32 - h; dy = -0.1;

direction = rand() % (3);

}

if (Dy < 0) {

y = i \* 32 + 32; dy = 0.1;

direction = rand() % (3);

}

if (Dx > 0) {

x = j \* 32 - w; dx = -0.1;

direction = rand() % (3);

}

if (Dx < 0) {

x = j \* 32 + 32; dx = 0.1;

direction = rand() % (3);

}

}

}

}

void *update*(float time)

{

if (name == "hardBoss"){

if (life) {

switch (direction)

{

case 0:{

dx = speed;

CurrentFrame += 0.005\*time;

if (CurrentFrame > 3) CurrentFrame -= 3;

sprite.setTextureRect(IntRect(96 \* int(CurrentFrame), 192, 96, 96));

break;

}

case 1:{

dx = -speed;

CurrentFrame += 0.005\*time;

if (CurrentFrame > 3) CurrentFrame -= 3;

sprite.setTextureRect(IntRect(96 \* int(CurrentFrame), 96, 96, 96));

break;

}

case 2:{

dy = -speed;

CurrentFrame += 0.005\*time;

if (CurrentFrame > 3) CurrentFrame -= 3;

sprite.setTextureRect(IntRect(96 \* int(CurrentFrame), 288, 96, 96));

break;

}

case 3:{

dy = speed;

CurrentFrame += 0.005\*time;

if (CurrentFrame > 3) CurrentFrame -= 3;

sprite.setTextureRect(IntRect(96 \* int(CurrentFrame), 0, 96, 96));

break;

}

}

x += dx\*time;

checkCollisionWithMap(dx, 0);

y += dy\*time;

checkCollisionWithMap(0, dy);

sprite.setPosition(x, y);

if (health <= 0){ life = false; }

}

}

}

}

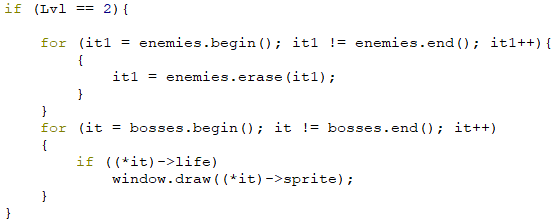
Создана текстура:

|  |  |
| --- | --- |
|  | Image hardBossImage;  hardBossImage.loadFromFile("images/boss.png");  …  for (int i = 0; i < BOSS\_COUNT; i++)  {  float xr = 300 + rand() % 500;  float yr = 300 + rand() % 350;  bosses.push\_back(new Boss(hardBossImage, xr, yr, 96, 96, "hardBoss"));  bossesCount += 1;  }  … |

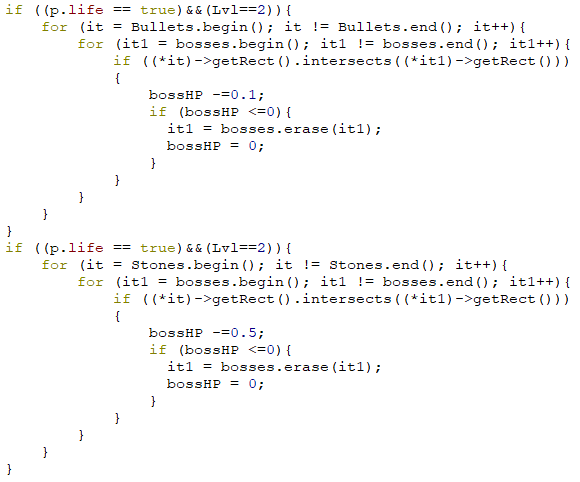
Прописано столкновение с объектами (игрок, фаерболл, камень)

|  |
| --- |
|  |
|  |
|  |

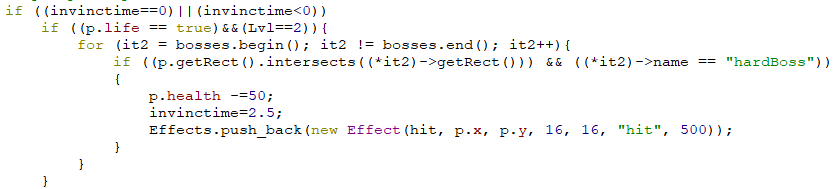
Прописано добавление босса и удаление мобов при переходе на второй уровень:



Реализовано нанесение урона камнем и фаерболлом боссу:



Реализовано нанесение повышенного урона игроку при столкновении с боссом:



Вывод: при разработке игры были добавлены несколько неотъемлемых элементов игры (стрельба, противник), доработаны некоторые логические ошибки (стрельба при диагональном передвижении, обработка событий разных уровней). Производилась работа с текстурами (создание спрайта, загрузка в игру). Была подробнее изучена контейнеризация (работа с контейнерами противников, снарядов).