ДИПЛОМНАЯ РАБОТА

Разработка 2D игр на C++

Руководитель

преподаватель,
Гладей Анатолий

Студент гр. I1802

Остапчук Максим

АКТУАЛЬНОСТЬ РАБОТЫ

- Разработка компьютерных игр - весьма вотребованная и актуальная профессия
- На сегодняшний день, разработка игры на языке не предоставляющем возможность использования ООП не является целесообразной
- C++ является актуальным языком в сфере разработки компьютерных игр
- 2D графика в современных играх всё ещё является востребованной

ЦЕЛИ РАБОТЫ

- Провести исследование, с целью изучить процесс разработки компьютерных игр
- Разработать компьютерную 2D игру в жанре “Tower Defence” с видом сверху на языке C++ с использованием мультимедийной библиотеки SFML

ЗАДАЧИ РАБОТЫ

- Создание 2D графики
- Реализовать взаимодействие игрока с игрой
- Реализовать работу объектно-ориентированной программы на взаимодействии и функционировании объектов различных классов .
- Разработать алгоритмы самостоятельного поведения объектов, в зависимости от происходящей обстановки .

ЗАМЫСЕЛ ИГРЫ

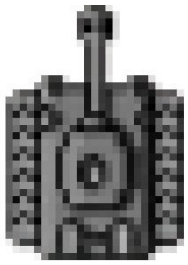
Игроку предстоит защитить свою базу от вражеского наступления путем расстановки оборонительных сооружений макимально эффективным образом , с целью получения наибольшего количества очков в финале игры.

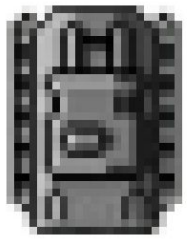
ВРАГИ

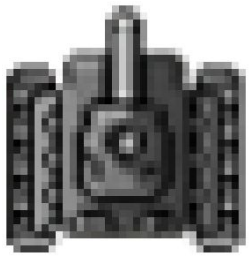
Враги представляют собой несколько волн вражеских единиц техники. Игрок должен, как можно эффективнее, предотвратить вражеское наступление

ВРАГИ

Враги	1-го типа	2-го типа	3-го типа
Количество	50%	30%	20%
Здоровье	30	50	150
Награда	25	50	100







ОРУЖИЯ

Из оборониельных сооружений, игроку доступны 2 вида оружия (турелей), которые отличаются своим функционалом.

Турель – представляет собой оружие с функцией самонаведения, которое автоматически наводится на врага . В зависимости от типа, каждая турель обладает различным функционалом по разному воздеёствует на врага.

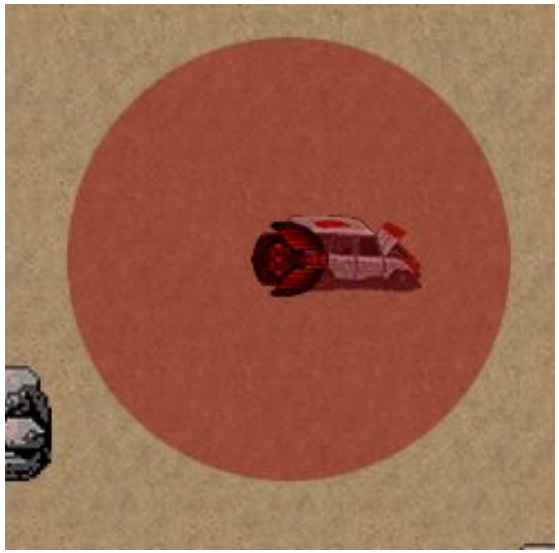
ОРУЖИЯ

Турели	1-го типа	2-го типа
Стоимость	50	150
Дальность	200	100
Функционал	Во время наведения, наносит врагу урон	Во время наведения, замедляет врага

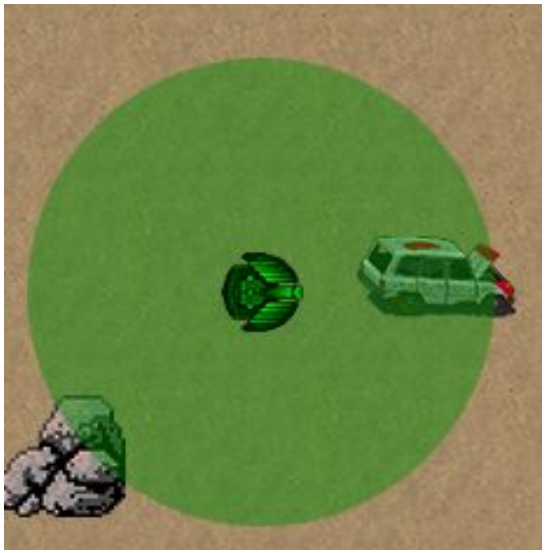
ВЗАИМОДЕЙСТВИЕ ИГРОКА С ИГРОЙ

- Игроку предоставляется возможность построить оборону вдоль маршрута врага таким образом, чтобы тот не прошёл к финальной точке своего маршрута.
- Игрок вправе распоряжаться валютой полученной при уничтожении единиц вражеской техники, которую он тратит на покупку турелей.
- Турели нельзя размещать на: дороге, физических объектах на карте .

ПРИМЕРЫ СОСТОЯНИЙ ТУРЕЛИ



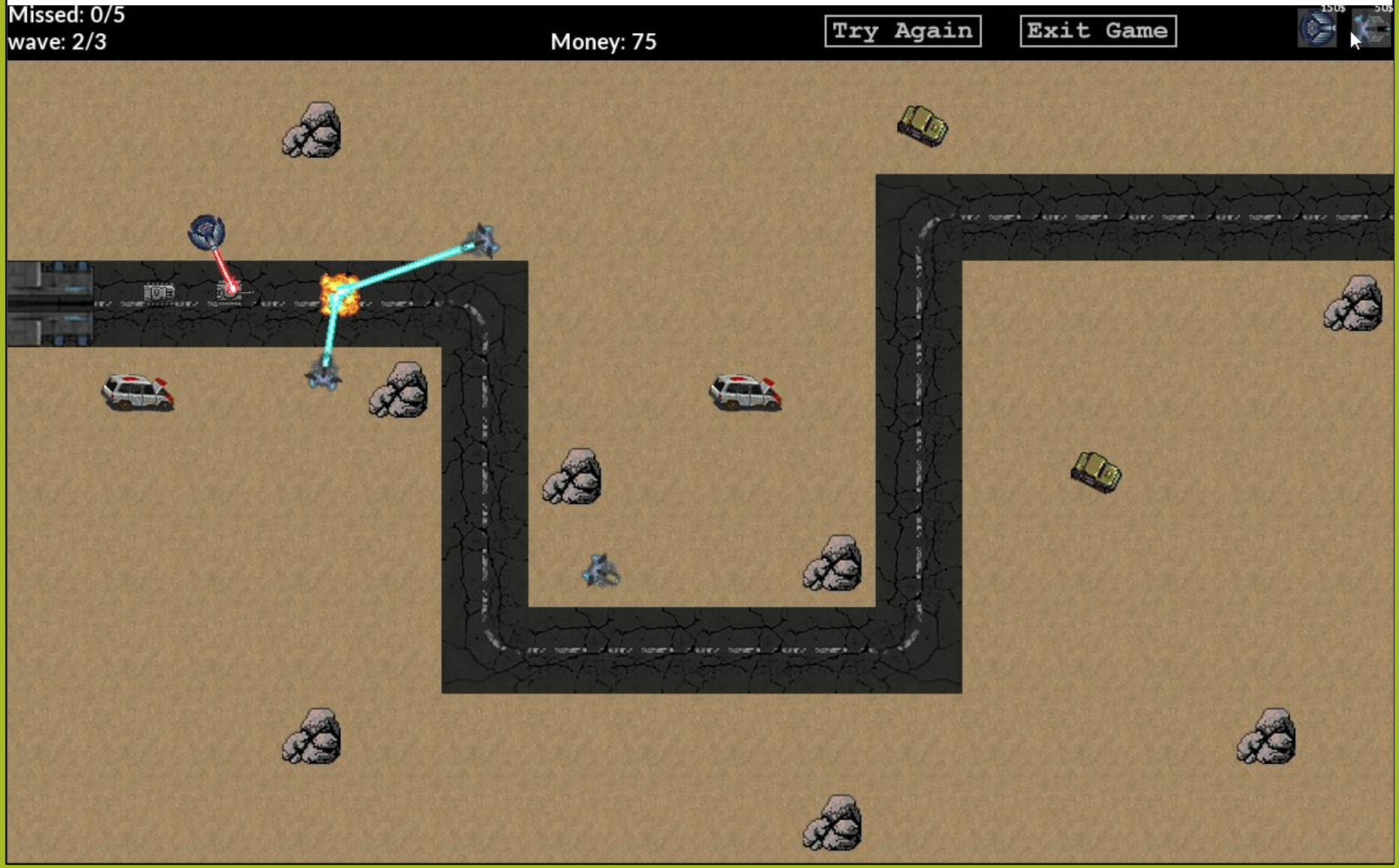
Турель не может быть размещена



Турель может быть размещена

Турель в действии

КАДР ИГРОВОГО ПРОЦЕССА



ПРИМЕР РАБОТЫ АЛГОРИТМА ИЗ ПРОГРАММЫ

Принцип работы алгоритма нахождения цели для турели.

- определение координат нахождения врага находящегося на максимальных координатах **x,y** в зоне действия турели
- Вычисление расстояния турели к врагу по **x** и по **y**
- Нахождение гипотенузы, как расстояния к врагу, по теореме пифагора
- нахождение угла наклона турели по формуле угла в прямоугольном треугольнике

Функция нахождения цели

```
//алгоритм определения цели обьекта.
void target(Arm& arm, Enemy en[], const int& k, float& CurrentFrame)
{
    int max_v = 0, max_f = 0; //значение расстояния максимально отдалённого врага и флаг, что такое найдено


    float a, b, c; //переменные используемые для решения уравнения о теореме пифагора, для вычисления гипотенузы(расстояния до врага)
                                //и для просчёта угла поворота турели

    if (arm.target == -1) {  //проверка, бездействует ли турель. -1 - значит у турели нет цели
        for (int i = 0; i < k; i++)
        {
            //по формуле вычисления гипотенузы, находим самый отталённый объект
            a = arm.getArmCoordinateX() - en[i].getEnemyCoordinateX(); // катет 1
            b = arm.getArmCoordinateY() - en[i].getEnemyCoordinateY(); // катет 2
            c = sqrt(pow(a, 2) + pow(b, 2)); // гипотенуза(расстояние до обьекта)

            //алгоритм поиска врага, проделавшего самый большой путь из всех и кторый находится в зоне действия турели
            if (c <= arm.range) {

                if (en[i].getEnemyCoordinateXY() > max_v)
                {
                    max_v = en[i].getEnemyCoordinateXY();
                    max_f = i;
                }
            }
            else { continue; }
        }
    }
    else {
        max_f = arm.target;
    }


    arm.Direction(en[max_f], CurrentFrame);  //задаём цель в метод наводки турели
}
```

Метод взаимодействия оружия с целью

```
// метод просчитывающий направление турели
void Direction(Enemy &en, float CurrentFrame )
{
    //задаём положение спрайта на карте
    sprite.setPosition(this->x, this->y);

    float a = 0, b = 0, c = 0; //переменные используемые для решения уравнения о теореме пифагора, для вычисления гипотенузы(расстояния до врага)
                               //и для просчёта угла поворота турели

    a = getArmCoordinateX() - en.getEnemyCoordinateX(); // катет 1
    b = getArmCoordinateY() - en.getEnemyCoordinateY(); // катет 2
    c = sqrt(pow(a, 2) + pow(b, 2)); // гипотенуза (расстояние до объекта)

    //уравнение для вычисления угла альфа, для получения значения поворота турели
    alph = acos((pow(c, 2) + pow(a, 2) - pow(b, 2)) / (2 * c*a)) * 180 / PI;

    target = -1; //значение, значит что у оружия пока нет цели

    //если расс. до объекта < 200 ( это значение зоны действия турели)
    if (c <= range)
    {
        target = en.getidEnemy();  //целью становится враг, который был "отправлен" в метод

        //в зависимости от того, где находится враг, под или над турелью, угол наклона меняет значение на 18градусов
        if (getArmCoordinateY() > en.getEnemyCoordinateY()) {
            sprite.setRotation(alph + 180);
            //вызываем метод лазер, который задаёт параметры длины расположения и угла наклона спрайту лазера
            Laser(c, alph, en.getEnemyCoordinateX(), en.getEnemyCoordinateY(), CurrentFrame, 1);
        }
        else {
            sprite.setRotation(-alph + 180);
            Laser(c, -alph, en.getEnemyCoordinateX(), en.getEnemyCoordinateY(), CurrentFrame, 1);
        }

        // в зависимости от типа оружия, на врага наносится конкретный эффект
        if (type == 1) {
            //тип 1: врагу наносится урон
            en.setDamage(0.2);
        }
        else if (type == 2) {
            //тип 2: враг замедляется
            en.setSpeedMultiplier(-0.06);
        }
    }
    else { // если выбранный враг не в зоне действия турели, оружие сообщает, что у него нет цели
        target = -1;
        Laser(c, -alph, en.getEnemyCoordinateX(), en.getEnemyCoordinateY(), CurrentFrame, 0); //вызываем метод лазер, чтоб задать значение не отображать лазер, пока турель простаивает
    }

}
```

ЗАКЛЮЧЕНИЕ

Обобщая результаты проведения исследовательской работы, можно утверждать, что для самостоятельной разработки игры необходимо обладать следующими навыками:

- планирование/структуризация масштабного проекта
- активная работа с ООП на протяжении всего цикла разработки
- работа со сторонними библиотеками
- разработка физической модели поведения врагов
- базовые знания математики

Спасибо за внимание!