Петрозаводский государственный университет
Институт математики и информационных технологий
Кафедра информатики и математического обеспечения

09.03.04 – Программная инженерия

Профиль направления подготовки бакалавриата

Системное и прикладное программное обеспечение

Отчет о проектной работе по курсу «Основы информатики и программирования»

Игра «Alien attack»

Выполнил:
студент 1 курса группы 22107

Д. А. Костин _____
подпись

Место прохождения практики:
Кафедра информатики и математического обеспечения

Период прохождения практики:

11.06.2021–11.06.2021

Руководитель:
А. В. Бородин, старший преподаватель

_____
подпись

Итоговая оценка:

_____
оценка

Петрозаводск — 2021

Содержание

Введение

Цель практики: Создать приложение с графическим интерфейсом

Задачи практики: Получить первый опыт работы с проектом, научиться составлять документацию, описывающую работу программы

Программа, позиционируемая как игра, - «Alien attack», - проект, созданный на языке Qml совместно с языком C++ в среде разработаны Qt Creator (с использованием имеющихся в нем библиотек), соответственно реализующие интерфейс и функциональные составляющие. Данная документация содержит всеобъемлющее описание как работы программы, так и математических и геометрических формул, используемых в программе при вычислениях.

1 Функциональная составляющая проекта

1.1 Модули на qml

1. Модуль Main.qml

Данный модуль является основой всех модулей на языке Qml:

- Определяет и фиксирует размеры окна программы,
- Загружает фон
- Обрабатывает нажатие мыши на область окна
- Задаёт свойства объектов игры

Для реализации взаимодействия модулей, были созданы глобальные свойства объектов (см. таблицу 1). Программа реагирует на любое нажатие левой кнопкой мыши в рабочем окне, что вызывает обработку нажатия: вычисляется вектор, направленный из центра самолёта в точку нажатия, координаты вектора передаются в функцию нормирования вектора - vector, также они передаются в функцию atang. Обе функции реализованы на C++, их рассмотрим позже.

Таблица 1 – Глобальные свойства модуля main.qml

Свойство	Функция
x_s | y_s	Отражают скорость полёта самолёта
r_x | r_y	Отражают скорость полёта ракеты
xold | yold	Являются координатами точки клика
reload	Количество секунд до перезарядки
score	Счёт сбитых НЛО
ready_strike	Флаг готовности пуска ракеты

2. Модуль Plane.qml

Данный модуль полностью посвящен игровому объекту - самолёту. Модуль загружает картинку самолёта, задаёт начальное положение самолёта и его поворот - таким образом его положение относительно фона - взлётная полоса, как будто он начинает с неё взлёт.

В модуле присутствует таймер, на каждый тик которого идёт проверка, что координаты самолёта достигли края - тут будет нужно применить его разворот обратно по правилу угла приломления из физики.

Если самолёт находится в пределах своего полёта, то на тик таймера к координатам самолёта прибавляются соответствующие глобальные свойства x_s и y_s.

3. Модуль Rocket.qml

Данный модуль похож на модуль Plane.qml, его задача - обработка движения выпущенной ракеты: каждый тик таймера к координатам ракеты прибавляются свойства r_x и r_y соответственно. Если ракета зашла за границу допустимой области, то её скорость приравнивается нулю.

4. Модуль GameFuncs.qml

Модуль GameFuncs определяет работу основных кнопок меню игры, находящихся внизу окна приложения: New_game и Quit. Из их названия их функционал очевиден. В том же меню находится строка счёта и количества секунд до перезарядки, которые рассчиты-

ваются здесь же через таймер с тиком каждую секунду.

Так как в данном модуле определены функции новой игры и выхода, то логично, что в данном модуле должна быть вызывающая их выбор функция - а именно функция поражения, представленная в листинге 1:

Листинг 1 – Код функции поражения

```
function lose(){
       var t = helper.rast(planer.x+45, planer.y+45, first.x+100, first.y+70)
       if (t[0] == true){
           planer.x = 2000
           planer.y = 2000
           losing.open()
       }
   }
```

Из кода видно, что сначала находится значение t - расстояние от центра самолёта до некоторого объекта - first - этот объект и есть НЛО. Если расстояние мало, то оператор if выполнит перенос самолёта за пределы экрана - тем самым имитируя уничтожение, открывается диалог с сообщением о крушении и выбором либо выхода из игры, либо началом новой игры.

5. Модуль Enemy.qml

Данный модуль реализует следующие функции:

- Получает случайную сторону и перемещает НЛО
- Двигает НЛО по таймеру
- Определяет попадание ракеты в НЛО и выпускает ракету

Пройдемся по функциям по порядку: перемещение в случайную точку в случайную сторону реализует функия rand на языке C++. Получив координаты, модуль на qml присваивает их к координатам НЛО. Сразу за этим, вызывается функция расчёта вектора

движения НЛО, также реализованная на C++.

Обратим внимание на работу таймера: каждый тик к координатам НЛО прибавляется скорость, вычисленная из функции расчёта вектора движения. Здесь же идёт расчёт расстояния от выпущенной ракеты до НЛО. Если их столкновение произошло, то ракета выводится за видимую область, и НЛО снова получает случайные координаты.

Отдельно рассмотрим, как происходит пуск ракеты. По нажатию клавиши «Space», происходит проверка, что ещё не прошла перезарядка или же, что самолёт попросту не начал движение. Оно и понятно - куда стрелять? Если же условия кода из листинга 2 выполнены, ракета переносится в центр самолёта, и получает скорость, равную удвоенной от самолёта. Флаг готовности к стрельбе становится false.

Листинг 2 – Код проверки готовности выпуска ракеты

```
if (ready_strike && x_s !=0 && y_s !=0)
{
    rok.x = planer.x + 45
    rok.y = planer.y + 45

    r_x = 2*x_s
    r_y = 2*y_s
    ready_strike = false
}
```

1.2 Модули на C++

Модуль helper.cpp

Единственный и основной модуль операций, проводимых на языке C++. Перед началом его рассмотрения, лучше перечислить все входящие в него функции:

- rast
- vector
- rand

- atang
- reject

И пусть читателя не пугает их не до конца ясное название. Далее мы разберем работу каждой подробно.

1. Функция rast

Получая на вход 4 параметра - соответственно две пары координат x и y. Не важно каких, главное возвратить результат, равный true, если расстоянии между двумя данными точками мало (менее 70 пикселей), или иначе false.
Данная формула максимально точно опишет работу данного модуля:

$$length = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2};$$

2. Функция vector

Получает на вход два параметра - координаты вектора, например, направления точки клика мыши. Для координат данного вектора применяются формулы нормирования вектора - то есть приведения его к единичной длине.

$$norgma_x = \frac{x}{\sqrt{x^2 + y^2}}; norgma_y = \frac{y}{\sqrt{x^2 + y^2}};$$

Делается это для того, чтобы домножив данные векторы на число получить правильную скорость движения объекта.

3. Функция rand

Данная же функция не получает на вход ни одного параметра, напротив, одна должна возвратить пару - координаты x и y, случайным образом сгенерированные программой (см. пример листинга 3) относительно базового числа - секунд от полночи до текущего времени.

Листинг 3 – Код функции поражения

```
QTime midnight(0,0,0);
qsrand(midnight.secsTo(QTime::currentTime()));
```

От данного сгенерированного числа берется остаток от деления на 4 - количество сторон, из которых может прилететь НЛО. Соответственно, если данный остаток 0 - значит НЛО прилетит сверху, 1 - слева, и так далее по часовой стрелке.

4. Функция atang

Функция получает два значения - координаты x и y, в программе этими координатами будут всё также координаты клика мыши. Программа сначала вычисляет тангенс через длины данных координат, затем по математической формуле с использованием тангенса вычисляет арктангенс - в градусах - который будет задавать поворот самолёта на место клика мыши. Ниже представлена используемая в программе формула.

$$rotation = \frac{\operatorname{arctg} \operatorname{tg} \frac{y}{x} * 180}{\pi};$$

Также в функции присутствует проверка на координатную четверть, так как отрицательный тангенс присутствует во 2 и 4 четвертях, но формула способна найти градусную меру лишь в полуокружности.

5. функция reject

Функция reject имеет задачу развернуть объект на некоторый угол, получаемый из того факта, что угол падения равен углу отражения. Однако здесь есть своя хитрость: рассчитывается не угол, который нужно добавить к текущему повороту, например, если тело падает под углом 30 градусов, то ему нужно повернуть ещё на 120 градусов, чтобы получился угол 150 (180 - 30 = 150), а рассчитывается угол, на который сам объект должен повернуться от положения в 0 градусов. То есть задача с падением в 30 градусов, в результате даст 180 - 30 = 150 - именно такой угол должно принять тело, не добавляя градусы к текущему повороту.

В функции присутствует проверка, что тело повёрнуто в сторону «отрицательных» градусов. В этом случае очевидно, что и поворт также должен получиться отрицательным, используется формула reject = -180 - x , где x текущий угол падения.

Заключение

Данное приложение получилось достаточно объемным в связи с большим количеством взаимодействий объектов таких, как ракета и НЛО, самолёт и НЛО, самолёт и граница окна и так далее. Решением вопросов взаимодействия модулей и обработки данных, стала инициализация глобальных свойств, к которым мог обратиться любой модуль для использования в своих функциях.

Неотъемлемой частью проекта стало совмещение языка Qml с C++. На языке C++ присутствует большая возможность математических вычислений, без которых было бы невозможно реализовать, например, случайное перемещение НЛО, после его уничтожение.

В проекте были продемонстрированы возможности языка Qml по созданию окна с рабочей областью мыши и панели с кнопками и строками вывода (например, счёт и секунды до перезарядки).