Проектная мастерская ГБОУ
«Бауманская инженерная школа №1580»

Создание трехмерного баллистического калькулятора с графическим интерфейсом

Направление «Информатика». Проектная группа ИП-20

Автор:
Крутиков Егор Александрович,
учащийся класса 10Д

Оглавление

Введение

Актуальность

Данный проект актуален как никогда. В условиях проведения СВО, потребности Министерства Обороны в качественном программном обеспечении достигают высокого уровня. Из-за экономических причин, все еще не доведена до конца цифровизация артиллерии. Многие батареи рассчитывают свои удары «по старинке»: с помощью бумаги, таблиц и линеек (ПУО-9).

Расчет по ПУО-9 хоть и является способом надежным, проверенным временем, но обладает рядом недостатков: временные затраты, перенос внимания на рутинные процессы, а не на артиллерийскую тактику.
Баллистический калькулятор решает данные проблемы, так как скорость вычислений многократно возрастает. Данный калькулятор можно будет легко портировать на мобильные устройства. Следовательно, можно сэкономить на специальных планшетах (СУО «Крапива»). Эти факторы делают проект не просто актуальным, но и действительно нужным в сегодняшних реалиях.

Также актуальность проекта по созданию трехмерного баллистического калькулятора с графическим интерфейсом может быть обоснована и другими факторами:

Спортивная стрельба:

В сфере спортивной стрельбы, особенно в дисциплинах, таких как стрельба из лука или стрельба из огнестрельного оружия, баллистические калькуляторы могут быть полезными для стрелков при расчете точности и учете внешних факторов.

Научные исследования:

В научных исследованиях баллистические калькуляторы могут применяться для моделирования различных сценариев, таких как поведение снарядов в атмосфере, что имеет значение для понимания физических законов и создания более эффективных боеприпасов.

Образование:

Проект может быть использован в образовательных целях для студентов, изучающих физику, инженерию или информатику. Это предоставляет возможность применить теоретические знания на практике и улучшить понимание взаимосвязи между различными дисциплинами.

Профессиональные области:

Инженеры, работающие в области аэрокосмической промышленности или разработке оружия, могут воспользоваться подобными инструментами для проектирования и тестирования новых систем.

Развлекательные цели:

Для энтузиастов стрельбы, охоты или просто любителей виртуальных симуляторов, трехмерный баллистический калькулятор с графическим интерфейсом может представлять интересную и познавательную игровую среду.

Таким образом, проект не только имеет практическое применение в различных областях, но также обеспечивает уникальную возможность интегрировать технологии информатики в решение реальных проблем.

Историческая справка

Баллистика, как наука о движении снарядов, имеет древние корни. Впервые вопросы баллистики стали активно изучаться в античности, когда армии стремились оптимизировать дальность и точность стрельбы из различных видов оружия.

В средние века и в Ренессанс математика стала играть все более важную роль в баллистике. Ученые разрабатывали формулы для расчета траекторий снарядов, учитывая воздействие силы тяжести и ветра.

В XIX веке с развитием промышленной революции появились первые механические калькуляторы, способные автоматически выполнять сложные математические операции. Они стали важными инструментами для баллистических расчетов.

В середине XX века с появлением компьютеров возможности в области баллистики значительно расширились. Разработка специализированных программ для расчета траекторий стала более доступной и эффективной. В последние десятилетия, с развитием графических технологий, программы для баллистических расчетов стали все более интерактивными и визуально привлекательными. Трехмерные модели и графические интерфейсы значительно облегчили восприятие результатов.

В настоящее время с ростом интереса к виртуальной реальности и трехмерной графике, проекты, такие как трехмерный баллистический калькулятор с графическим интерфейсом, становятся актуальными инструментами для визуализации и практического применения баллистических расчетов.

Этот проект, сочетая в себе традиции баллистики и современные технологии, может быть рассмотрен как следующий шаг в эволюции инструментов для анализа и моделирования траекторий снарядов.

Цель проекта

Разработка трехмерного баллистического калькулятора с графическим интерфейсом для моделирования и визуализации траекторий полета снарядов с учетом различных физических параметров.

Задачи проекта

1. **Проектирование интерфейса**

Разработать удобный и интуитивно понятный графический интерфейс пользователя для ввода начальных данных, таких как начальная скорость, угол запуска и другие физические параметры.

2. **Математическое моделирование**

Разработать математический алгоритм для расчета траектории снаряда, учитывая воздействие силы тяжести, аэродинамические сопротивление и другие факторы.

3. **Трехмерная визуализация:**

Использовать современные библиотеки и технологии для трехмерной визуализации, предоставляя пользователю наглядное представление траектории полета снаряда.

4. **Учет внешних факторов**

Внедрить возможность учета внешних факторов, таких как ветер, гравитация, и изменяемые параметры среды, чтобы сделать расчеты более реалистичными.

5. **Гибкость настроек**

Предоставить пользователю широкий спектр настроек, позволяющих изменять параметры снаряда, окружающей среды и другие факторы для моделирования различных сценариев.

6. **Оптимизация производительности**

Обеспечить оптимальную производительность программы, особенно при работе с большим объемом данных для трехмерной визуализации.

7. **Тестирование и отладка**

Провести систематические тесты для проверки корректности расчетов и графической визуализации, а также устранить возможные ошибки.

8. **Документация и руководство пользователя**

Подготовить подробную документацию, объясняющую принцип работы программы, алгоритмы расчетов и предоставляющую руководство пользователя.

9. **Эргономика и дизайн**

Обеспечить хорошую эргономику и дизайн интерфейса, чтобы пользователь мог легко ориентироваться в приложении и использовать его без затруднений.

10. **Развитие дополнительных функций**

По возможности, добавить дополнительные функции, такие как сохранение и загрузка данных, анимации движения снаряда и другие элементы, повышающие интерес и возможности пользователя.

Эти задачи представляют собой комплексный подход к разработке проекта, объединяя в себе аспекты программирования, математического моделирования и дизайна пользовательского интерфейса.

Аналоги

Существует несколько аналогичных проектов и программ, которые предоставляют функциональность по расчету траекторий снарядов с графическим интерфейсом. Некоторые из них ориентированы на военные или научные цели, в то время как другие могут быть использованы для образовательных или развлекательных целей. Вот несколько примеров:

Strelok Pro

Цель: Расчет баллистических характеристик для стрелков и охотников.
Особенности: Предоставляет расчеты для различных видов огнестрельного оружия, учет ветра и других факторов.

Applied Ballistics

Цель: Профессиональные баллистические расчеты для стрелков и снайперов.
Особенности: Включает сложные алгоритмы для учета ветра, влияния кориолисова эффекта, а также возможность создания пользовательских профилей.

VirtualBow

Цель: Моделирование стрельбы из лука.
Особенности: Позволяет стрелкам тренироваться в виртуальной среде, учитывая физические параметры стрелы и динамику полета.

Ballistic AE

Цель: Расчет баллистических данных для охоты и стрельбы.
Особенности: Предоставляет подробные графики траекторий, а также возможности для учета изменяемых условий.

Wolfram Demonstrations Projects: 3D Projectile Motion

Цель: Визуализация траекторий снарядов в трехмерном пространстве.
Особенности: Позволяет пользователям вводить параметры снаряда и окружающей среды, а затем наблюдать траекторию в трехмерной модели.

Эти аналоги представляют собой разнообразие подходов к решению задачи расчета баллистических траекторий, они различаются по уровню сложности, целевой аудитории и функциональности.

Основная часть

Построение физической модели

В рамках проекта по созданию трехмерного баллистического калькулятора с графическим интерфейсом, необходимо разработать физическую модель, которая будет основой для расчета траектории полета снаряда. Эта глава описывает ключевые аспекты построения физической модели, учитывающей основные факторы, влияющие на движение снаряда в трехмерном пространстве.

1. Моделирование движения

Для построения физической модели необходимо учитывать основные физические законы, определяющие движение снаряда. Основные параметры, которые будут включены в модель, включают:

- Начальная скорость (v_0): Скорость снаряда при покидании ствола.
- Угол вертикальной наводки (α): Угол между направлением начальной скорости и горизонтом.
- Угол горизонтальной наводки (β): Угол между направлением начальной скорости и осью Ox (север).
- Масса снаряда (m): Масса снаряда, влияющая на его инерцию и взаимодействие с внешней средой.

Скорость будет представлять собой трехмерный вектор $\vec{v}\{x, y, z\}$.

$$\vec{v}\{v\cos\alpha\cos\beta, v\cos\alpha\sin\beta, v\sin\alpha\}$$

2. **Учет сопротивления воздуха**

Добавим учет воздушного сопротивления для более реалистичного моделирования. Так как скорости моделируемых тел предполагаются большими, сила сопротивления будет пропорциональна квадрату скорости. Силу воздушного сопротивления можно представить в виде:

$$\vec{F_s} = -\mu|v_{rel}|\vec{v_{rel}}$$

Где μ — некий аэродинамический коэффициент, определяющийся характеристиками снаряда, а $\vec{v_{rel}}$ — скорость снаряда относительно воздушной среды.

3. **Скорость ветра**

Так как воздушная среда в реальных условиях почти никогда не бывает стоячей, необходимо ввести в калькулятор учет скорости ветра (а также его направления). Представим $\vec{u}$ в виде двух компонент:

- Модуль скорости ветра (u)
- Направление ветра относительно оси Ox (γ)

Тогда

$$\vec{u}\{u\cos\gamma, u\sin\gamma, 0\}$$

Относительная скорость в произвольный момент времени:

$$\vec{v_{rel}} = \vec{v} - \vec{u}$$

4. **Уравнение движения**

Запишем уравнение движения для снаряда, используя II закон Ньютона, ускорение свободного падения и приведенные выше формулы:

$$\vec{a} = \vec{g} - \frac{\mu|\vec{v_{rel}}|\vec{v_{rel}}}{m}$$

Где $\vec{g}\{0, 0, -9.81\}$ — ускорение свободного падения
Параметры будут изменяться по следующим законам:

$$\begin{cases} d\vec{v} = \vec{a} \cdot dt \\ d\vec{r} = \vec{v} \cdot dt \end{cases} \quad (1)$$

Где $\vec{r}$ — радиус-вектор из начала координат в точку текущего нахождения снаряда

Интеграция в программное обеспечение

Физическая модель, построенная на основе уравнений движения и учета воздушного сопротивления, будет реализована на языке программирования C++. Использование C++ позволяет эффективно оптимизировать код для обеспечения быстрых расчетов и точных результатов. Чтобы хранить данные и вычислять векторные выражения, создадим инструментарий трехмерной вычислительной геометрии:

Класс `P(float x, float y, float z)` — точка,
Класс `AVec(float vx, float vy, float vz)` — свободный вектор,

Класс **Vec**(P p1, P p2) — геометрический вектор.

Перегрузим операторы сложения и умножения для новых классов:

```
P operator + (P p1, AVec av){
   return P(p1.x + av.x, p1.y + av.y, p1.z + av.z);
}

AVec operator + (AVec av1, AVec av2){
   return AVec(av1.x + av2.x, av1.y + av2.y, av1.z +
av2.z);
}

AVec operator * (AVec av, float k){
   return AVec(av.x*k, av.y*k, av.z*k);
}

Vec operator * (float k){
   return Vec(p1, p1 + to_avec()*k);
}
```

А также добавим некоторые вспомогательные методы, например, вычисление длины вектора и конвертация геометрического вектора в свободный:

```
float length(){
   return sqrt(pow(x, 2) + pow(y, 2) + pow(z, 2));
}

AVec to_avec(){
       return AVec(p2.x - p1.x, p2.y - p1.y, p2.z -
p1.z);
}
```

Таким образом, инструментарий для работы с векторами в трехмерном пространстве создан. Следующая задача — сделать подпрограмму, которая будет принимать начальные параметры, вычислять симуляцию, и отдавать все необходимые конечные значения. Оформим её в функцию

```
Simulation compute(Vec v0, Vec u, float mu, float m,
float dt)
```

Где v0 — вектор начальной скорости, u — скорость ветра, mu — коэффициент аэродинамического сопротивления, m — масса.

Так как получить аналитическое решение системы дифференциальных уравнений (1) не представляется возможным, для нахождения параметров симуляции будет использован метод численного интегрирования. Этот метод позволяет приближенно вычислить траекторию снаряда на каждом шаге времени, учитывая изменение физических параметров, таких как скорость и положение. Суть метода численного интегрирования заключается в приближенном вычислении интегралов дифференциальных уравнений, описывающих движение снаряда. Для этого дополнительно задаем параметр dt — шаг интегрирования.

Чтобы хранить данные симуляции на выходе из функции (для дальнейшего их отображения пользователю), создадим структуру

```
struct Simulation{
    QScatterDataArray data;
    float z_max;
    AVec v_end;
};
```

Где data — массив точек, в которых побывало тело во время полета, z_max — максимальная высота снаряда, v_end — конечная скорость.

В итоге, получаем такую функцию для расчётов:

```
Simulation compute(Vec v0, Vec u, float mu, float m,
float dt){
    P r = P();
    AVec v = v0.to_avec();
    AVec v_;
    QScatterDataArray data;
    float z_max = 0;

    do{
        r = r + v*dt;  // Вычисляем дифференциал радиус
вектора
        v_ = v + u.to_avec()*(-1); // Скорость
относительно воздуха (ветра)
        v = v + (G + v_*(-mu*v_.length()/m))*dt; //
Вычисляем дифференциал скорости

        data << QVector3D(r.x, r.z, r.y); // Особенности
Q3DScatter (y -> z)

        if (r.z > z_max){ // h_max
            z_max = r.z;
        }

    } while(r.z > 0);

    return Simulation{data, z_max, v};
}
```

Создание графического интерфейса

Интерфейс служит средством взаимодействия пользователя с физической моделью и обеспечивает легкий доступ к функциональности калькулятора. Поэтому, интерфейс должен быть удобным и простым. В качестве фреймворка для создания приложения был выбран Qt, так как в нем можно одновременно и писать код, и заниматься дизайном, а также, он поддерживает разработку на языке C++, на котором была написана вычислительная часть программы.

1. Оконный дизайн

При создании GUI на Qt, первоначальным шагом является определение оконного дизайна. Фреймворк Qt предоставляет инструменты для создания главного окна и дополнительных диалоговых окон, а также для добавления виджетов, таких как кнопки, поля ввода и элементы управления.

2. Поля ввода

Входные параметры, задаваемые количественно (такие как скорость, коэффициент сопротивления, масса и шаг интегрирования) будут вводиться в объекты типа LineEdit (однострочный простой ввод)

Для значений углов сделаем более интуитивный ползунок, отклонение которого в одну из сторон будет давать на выходе +180°, а в другую — -180°. Таким способом, получится подавать на вход углы в очень широких диапазонах, однако, точность из-за этого резко падает. Чтобы это исправить, добавим рядом с ползунком кнопку-маркер,

позволяющую перейти в более точный режим с диапазоном $[x - 4.5°, x + 4.5°]$.

3. Вывод

Сделаем поля вывода с помощью объектов типа Label. Среди них будут:

- Координаты падения в формате (--, --)
- Время полета
- Дальность полета
- Максимальная высота
- Конечная скорость
- Конечный угол

4. Компоновка

Скомпануем поля ввода и вывода в окне графического интерфейса, добавим к ним кнопку «Расчет» и место под будущий трехмерный график траектории. Результат показан в **Приложении 1**.

Трехмерный график траектории

Для достижения высокой степени наглядности и визуальной привлекательности, в проекте будет использоваться трехмерный график, Qt предоставляет инструмент Q3DScatter для реализации этой задачи. Q3DScatter - это класс библиотеки Qt Data Visualization, предназначенный для построения трехмерных графиков на основе различных данных. В контексте нашего проекта, Q3DScatter идеально

подходит для визуализации точек траектории снаряда в трехмерном пространстве. Для внедрения трехмерного графика в графический интерфейс, мы интегрируем Q3DScatter как виджет в приложение Qt. Это обеспечивает взаимодействие пользователя с трехмерной визуализацией наряду с остальными элементами графического интерфейса.

1. Передача данных в Q3DScatter

Q3DScatter принимает данные в виде массивов точек в трехмерном пространстве. Эти данные могут быть получены из результатов численного интегрирования физической модели. За это отвечает массив `QScatterDataArray data` в структуре результатов симуляции `struct Simulation`.

Код, который отвечает за добавление точек на график, приведен в **Приложении 2**.

2. Внешний вид и взаимодействие

Qt Data Visualization предоставляет широкие возможности для настройки внешнего вида трехмерных графиков. Мы можем регулировать цвет точек, добавлять метки и оси, а также регулировать масштаб и углы обзора, чтобы обеспечить лучшую наглядность визуализации. Q3DScatter обеспечивает интерактивность для пользователя, позволяя вращать, масштабировать и перемещать график. Это не только улучшает пользовательский опыт, но также дает возможность более детального изучения траектории.

В баллистике часто приходится сравнивать траектории двух или более выстрелов. Для реализации этой возможности введем кнопку-маркер «Фиксировать график», которая вместо удаления предыдущего графика будет менять его цвет, оставляя его доступным пользователю.

3. Оптимизация

При работе с большим объемом данных о трехмерных точках важно помнить о производительности. Qt Data Visualization предоставляет средства для эффективной отрисовки трехмерных графиков.

Таким образом, использование Q3DScatter в проекте позволяет создать эффективную и наглядную трехмерную визуализацию траектории снаряда, делая баллистический калькулятор более привлекательным и понятным для пользователя.

Подготовка к использованию

После завершения разработки трехмерного баллистического калькулятора на фреймворке Qt с использованием C++, необходимо подготовить приложение для использования на других машинах. Этот процесс включает в себя создание исполняемого файла (.exe), который можно запускать на целевой системе без необходимости установки среды разработки.

Qt предоставляет инструментарий для развертывания приложений, так что остается только собрать проект в режиме Release и воспользоваться

утилитой windeployqt.exe, которая добавит необходимые библиотеки Qt в папку с исполняемым файлом.

1. GitHub

Добавление репозитория с программой на GitHub обеспечивает доступность проекта. Также, это позволяет удобно контролировать изменения в программной части.

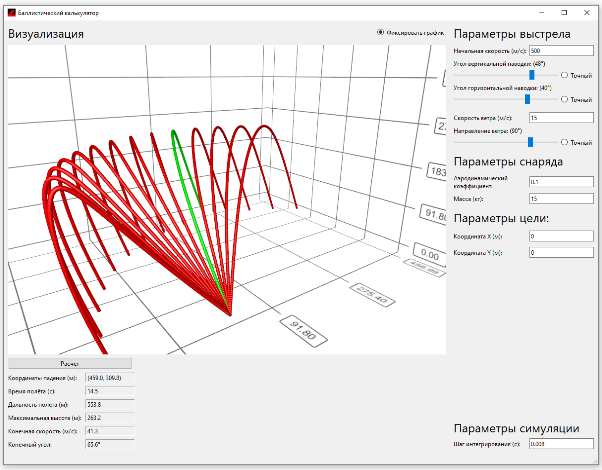
Вот ссылка на репозиторий с исходным кодом:
https://github.com/spaceshine/Ballistic-calculator

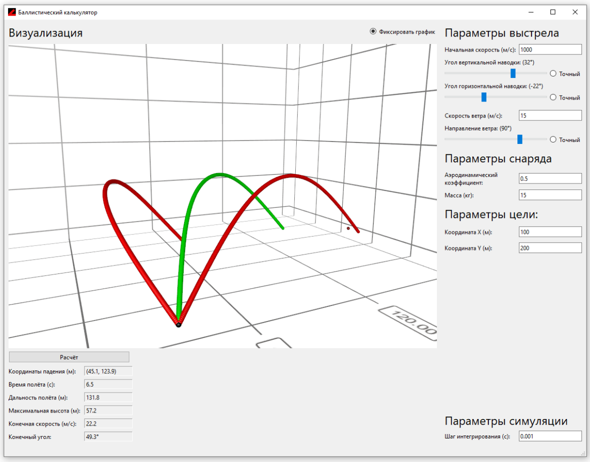
Также, туда можно добавить папку с собранной программой:
https://github.com/spaceshine/Ballistic-calculator/releases/tag/Release

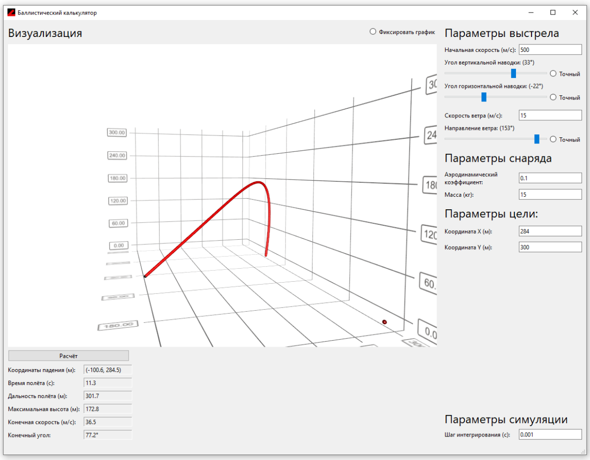
2. Инструкция по использованию

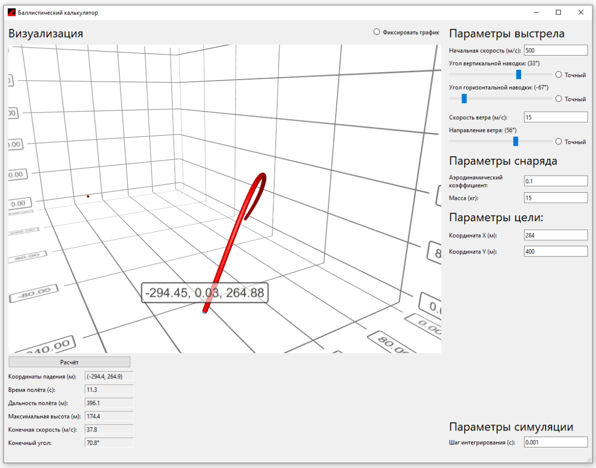
Для еще большей доступности и понятности баллистического калькулятора стоит сделать краткую инструкцию, которая будет объяснять как пользоваться программой. Получившаяся картинка прикреплена в **Приложении 3**.

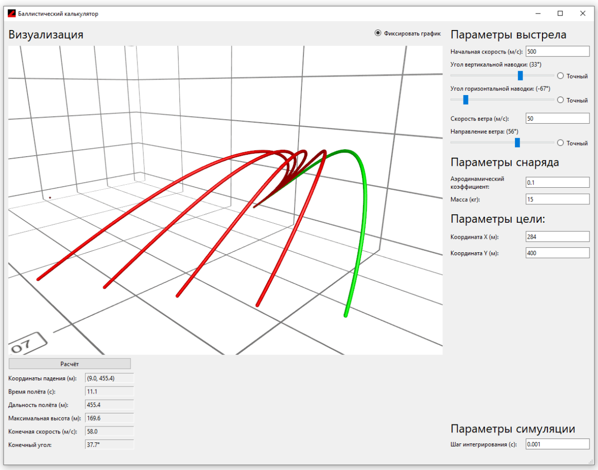
Использование

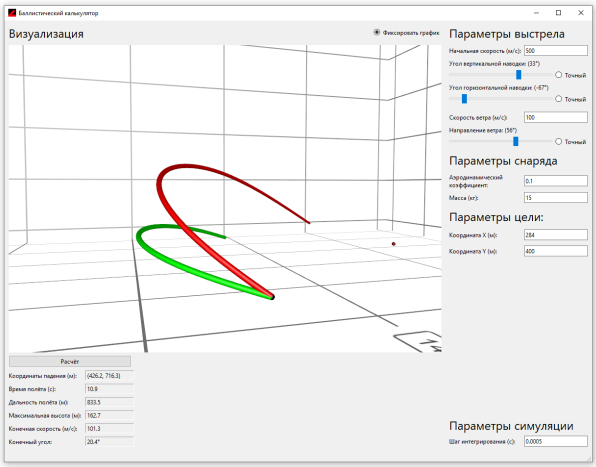






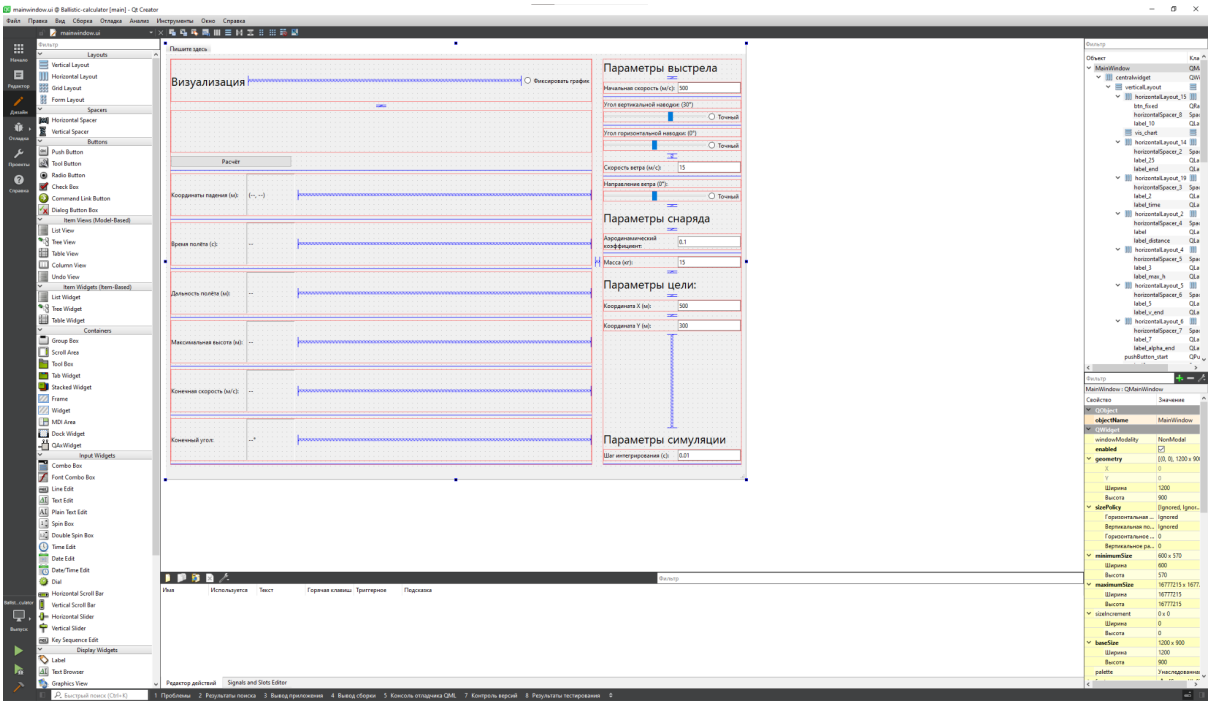






Приложения

Приложение 1



Приложение 2

```
// Убираем предыдущий график
if (! ui->btn_fixed->isChecked()){
    for (auto ser : chart->seriesList()){
        chart->removeSeries(ser);
    }
} else {
    chart->seriesList().at(0)->setBaseColor(Qt::green);
}

// Добавляем точки
series = new QScatter3DSeries;
series->dataProxy()->addItems(result.data);
series->setBaseColor(Qt::red);
chart->addSeries(series);
```

```
// Добавляем точку старта на график отдельным цветом
start_point = new QScatter3DSeries;
QScatterDataArray start_p_data;
start_p_data << QVector3D(0.0, 0.0, 0.0);
start_point->setBaseColor(Qt::black);
start_point->setItemSize(series->itemSize()*1.2f);
start_point->dataProxy()->addItems(start_p_data);
chart->addSeries(start_point);

// Добавляем точку цели на график отдельным цветом
target_point = new QScatter3DSeries;
QScatterDataArray target_p_data;
target_p_data << QVector3D(target_y, 0.0, target_x);
target_point->setBaseColor(Qt::darkRed);
target_point->setItemSize(series->itemSize()*1.2f);
target_point->dataProxy()->addItems(target_p_data);
chart->addSeries(target_point);

// Настраиваем оси и показываем график
float new_range = std::max(std::max(abs(end.x),
abs(end.y)), abs(result.z_max));
new_range = std::max(std::max(abs(target_x),
abs(target_y)), new_range);
if (ui->btn_fixed->isChecked()){
   range = std::max(range, new_range);
} else {
   range = new_range;
}
chart->axisX()->setRange(-range, range);
chart->axisY()->setRange(0, range);
chart->axisZ()->setRange(0, range);
chart->setAspectRatio(2);
chart->setHorizontalAspectRatio(2);
chart->show();
```

Приложение 3