THR: Бросок под углом к горизонту

Математическое моделирование: динамика.

Рассчитайте траекторию полёта снаряда, выпущенного из пушки, стоящей в начале координат, под углом $\alpha_0 = 60^\circ$ со скоростью $v_0 = 20$ м/с.

0. **Нарисуйте мировые оси (0 баллов).** Установите соответствие между экраном и мировыми координатами $[-3, 41]$ метров по x (вправо) и $[-5, 28]$ метров по y (вверх), соотношение сторон подобрано 4 к 3, как и у экрана, чтобы не было искажений. Для этого заведите структуры `point_t` для вещественной точки, `rect_t` для прямоугольной области, а также функцию

   ```
   point_t Transform(point_t p, rect_t const* from, rect_t const* to);
   ```

   которая преобразует координаты точки из одной произвольной прямоугольной области в другую, возвращая новые координаты. Напишите функцию

   ```
   void DrawAxes(rect_t const* math, rect_t const* screen);
   ```

   в которой возьмите центр математических координат (ноль) и переведите его в экранные, а затем в полученной точке нарисуйте оси.

1. **Отобразите аналитическую траекторию (+1 балл).** Из курса физики известно, что идеальной траекторией материальной точки при постоянном ускорении в вакууме является парабола. В векторной форме её можно записать как

   $$\vec{r} = \vec{r}_0 + \vec{v}_0 t + \frac{1}{2}\vec{a}_0 t^2,$$

   где $\vec{r}_0, \vec{v}_0, \vec{a}_0$ — начальное положение, скорость и ускорение (свободного падения в данном случае). Напишите функцию для рисования траектории **зелёным** цветом по этой заранее выведенной формуле.

   ```
   void DrawAnalyticalPath(rect_t const* math, rect_t const* screen,
     vec_t r0, vec_t v0, vec_t a0, double dt);
   ```

   Для этого, меняя в цикле время t_k от $t_0 = 0$ до $t_{\max} = 2v_{0,y}/a_{0,y}$ с шагом $\Delta t = 1$ с, соедините отрезками получающиеся по формуле точки $\vec{r}_k$. Такую же кривую постройте с шагом 0.5 с — они будут постоянно соприкасаться, так как решение точное.

2. **Вычислите “реальную” траекторию (+1 балл).** Найдите траекторию материальной точки при помощи имитационного моделирования, учитывая действующие на тело силы. Плюс этого метода в том, что численно можно решать более сложные уравнения и находить траектории движения под воздействием большого количества нетривиальных сил. Аналитическое решение в этих случаях построить зачастую невозможно.

   Вспомните закон Ньютона

   $$m\vec{a} = \vec{F}.$$

Ускорение в каждый момент времени пропорционально сумме всех сил. Силы в общем случае могут зависеть от времени, координаты или скорости. Запишем всё через координаты точки. Ускорение — это вторая производная, скорость — первая, поэтому получаем следующее *дифференциальное уравнение* второго порядка относительно координаты (если вы ещё не умеете их решать, пусть вас это не пугает).

$$\frac{d^2\vec{r}}{dt^2} = \frac{1}{m}\vec{F}(t, \vec{r}, \frac{d\vec{r}}{dt}),$$

Уравнение второй степени принято заменять на систему из двух уравнений первой степени, введя вспомогательную переменную для первой производной (как это ни странно, назовём её $\vec{v}$).

$$\frac{d\vec{r}}{dt} = \vec{v}, \qquad \frac{d\vec{v}}{dt} = \frac{1}{m}\vec{F}(t, \vec{r}, \vec{v}).$$

Для решения этой системы (получения одного шага решения) будем использовать грубый метод Эйлера, который производные приближает конечными приращениями:

$$\Delta\vec{r}_k = \vec{v}_k \cdot \Delta t_k, \qquad \Delta\vec{v}_k = \frac{1}{m}\vec{F}(t_k, \vec{r}_k, \vec{v}_k) \cdot \Delta t_k.$$

Имея эти приближённые формулы и начальные значения t_0, $\vec{r}_0$, $\vec{v}_0$, можем в цикле дойти до любого момента времени. Напишите функцию для рассчёта и отображения траектории по методу Эйлера **красным** цветом для $\Delta t = 1$ с и 0.5 с (пусть траектория останавливается, если тело провалилось под землю):

```
void DrawEulerPath(rect_t const* math, rect_t const* screen,
  vec_t r0, vec_t v0, vec_t a0, double dt);
```

Траектория выходит не только ломаной, но и неточной, потому что алгоритм Эйлера имеет большую погрешность. Чем больше шаг, тем менее точное решение (есть методы и получше, напр. Рунге-Кутты).

3. **Учтите реальное время (+1 балл)**. Иногда при моделировании важен не только конечный ответ (где окажется материальная точка?), но и реалистичность всего процесса — например, в системах виртуальной реальности, тренажёрах, компьютерных играх. Скопируйте и измените функцию так, чтобы снаряд летел реалистично, синхронно с течением времени нашего реального мира (**жёлтым** цветом). Полёт займёт чуть больше 3.5 секунд.

```
void SimulateEulerPath(rect_t const* math, rect_t const* screen,
  vec_t r0, vec_t v0, vec_t a0);
```

В начале моделирования запомните текущее время. На каждом шаге цикла также опрашивайте таймер, вычисляйте, сколько прошло времени с предыдущей итерации. Используйте это приращение в качестве Δt, и вы будете наблюдать полёт снаряда в реальном времени. Используйте `QueryPerformanceCounter` / `...Frequency` из `windows.h`.