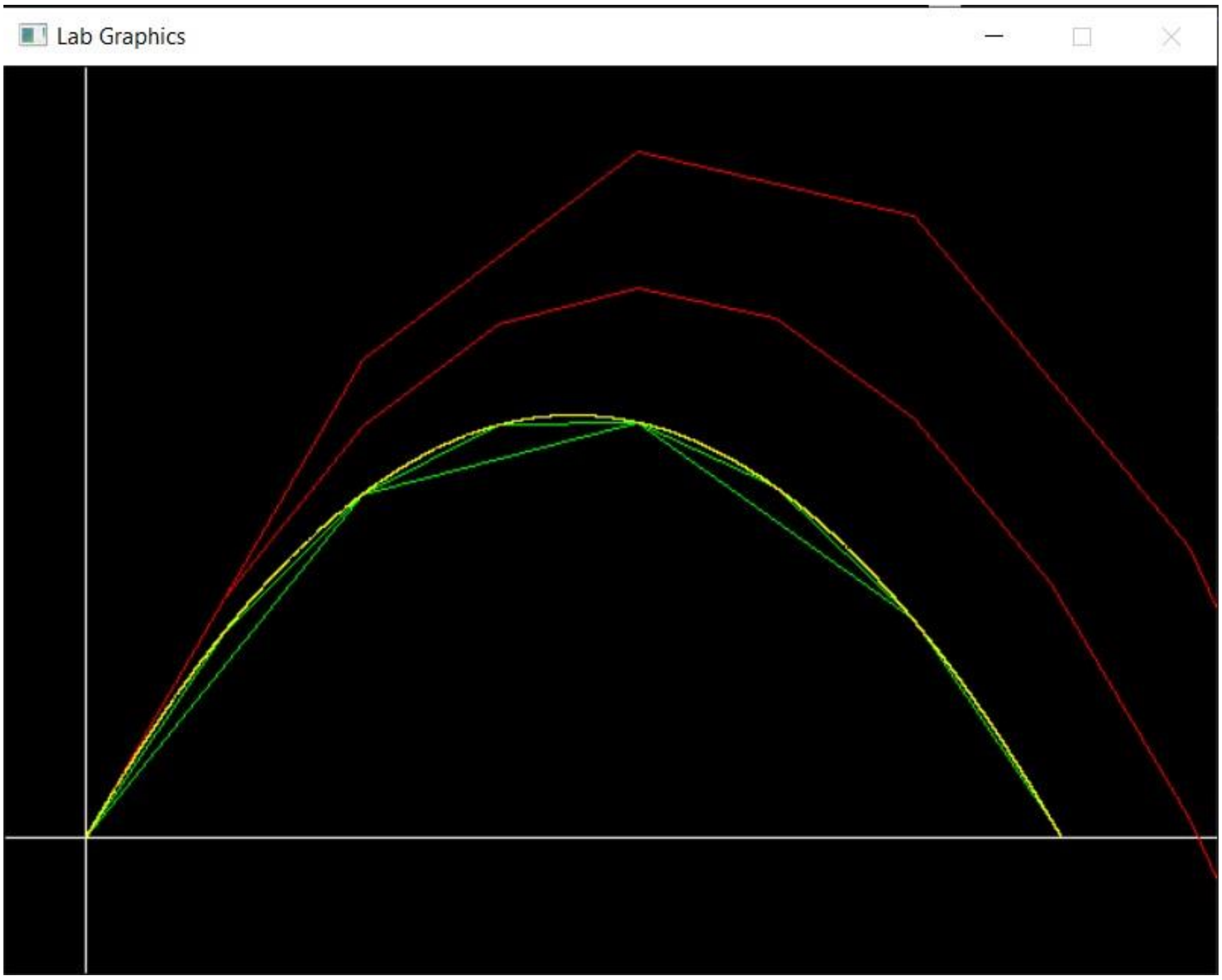
THR: Бросок под углом к горизонту

Код

```
#include <math.h>
#include <stdio.h>
#include <windows.h>

#include "labengine.h";

#define PI 3.1415

typedef struct {
    double x;
    double y;
} point_t;

typedef point_t vec_t;

typedef struct {
    point_t lt;
    point_t rb;
} rect_t;


point_t Transform(point_t p, rect_t const* from, rect_t const* to) {
    point_t q;
    q.x = (p.x - from->lt.x) * (to->rb.x - to->lt.x) / (from->rb.x - from->lt.x)
+ to->lt.x;
    q.y = (p.y - from->lt.y) * (to->rb.y - to->lt.y) / (from->rb.y - from->lt.y)
+ to->lt.y;
    return q;
}


void DravvAxes(rect_t const* math, rect_t const* screen) {
    point_t math_zero;
    math_zero.x = 0.;
    math_zero.y = 0.;
    point_t screen_zero = Transform(math_zero, math, screen);

    LabDrawLine(screen_zero.x, screen->lt.y + 1, screen_zero.x, screen->rb.y -
1);
    LabDrawLine(screen->lt.x + 1, screen_zero.y, screen->rb.x + 1,
screen_zero.y);
}


void DravvAnalyticalPath(
    rect_t const* math, rect_t const* screen,
    vec_t r0, vec_t v0, vec_t a0, double dt
) {
    double t_max = -2 * v0.y / a0.y;
    vec_t r_current = r0;

    for (double t_current = 0.; t_current < t_max; t_current += dt) {
        vec_t r_next;
        r_next.x = r0.x + v0.x * t_current + t_current * t_current * a0.x / 2;
        r_next.y = r0.y + v0.y * t_current + t_current * t_current * a0.y / 2;

        vec_t r_current_screen = Transform(r_current, math, screen);
        vec_t r_next_screen = Transform(r_next, math, screen);
        LabDrawLine(r_current_screen.x, r_current_screen.y, r_next_screen.x, 
r_next_screen.y);
        r_current = r_next;
    }
}


void DravvEulerPath(
    rect_t const* math, rect_t const* screen,
    vec_t r0, vec_t v0, vec_t a0, double dt
) {
    vec_t v_current = v0;
    vec_t r_current = r0;

    while (r_current.y >= 0) {
        vec_t r_next;
        r_next.x = r_current.x + v_current.x * dt;
        r_next.y = r_current.y + v_current.y * dt;
        v_current.y += a0.y * dt;

        vec_t r_current_screen = Transform(r_current, math, screen);
        vec_t r_next_screen = Transform(r_next, math, screen);
        LabDrawLine(r_current_screen.x, r_current_screen.y, r_next_screen.x, 
r_next_screen.y);
        r_current = r_next;
    }

}


void SimulateEulerPath(
    rect_t const* math, rect_t const* screen,
    vec_t r0, vec_t v0, vec_t a0
) {
    vec_t v_current = v0;
    vec_t r_current = r0;

    LARGE_INTEGER frequency;
    LARGE_INTEGER prevTime, currentTime;
    QueryPerformanceFrequency(&frequency);
    QueryPerformanceCounter(&prevTime);
    while (r_current.y >= 0) {
        QueryPerformanceCounter(&currentTime);
        double dt = (double)(currentTime.QuadPart - prevTime.QuadPart) / 
frequency.QuadPart;
        prevTime = currentTime;

        vec_t r_next;
        r_next.x = r_current.x + v_current.x * dt;
        r_next.y = r_current.y + v_current.y * dt;
        v_current.y += a0.y * dt;

        vec_t r_current_screen = Transform(r_current, math, screen);
        vec_t r_next_screen = Transform(r_next, math, screen);
```

```c
        LabDrawLine(r_current_screen.x, r_current_screen.y, r_next_screen.x,
r_next_screen.y);
        r_current = r_next;
    }

}


int main(void) {
    if (LabInit()) {
        point_t lt_math, rb_math, lt_screen, rb_screen;
        lt_math.x = -3.;
        lt_math.y = 28.;
        rb_math.x = 41.;
        rb_math.y = -5.;

        lt_screen.x = 0.;
        lt_screen.y = 0.;
        rb_screen.x = LabGetWidth();
        rb_screen.y = LabGetHeight();

        rect_t math, screen;
        math.lt = lt_math;
        math.rb = rb_math;

        screen.lt = lt_screen;
        screen.rb = rb_screen;

        DravvAxes(&math, &screen);

        double alpha = 60;
        double alpha_rad = 60 * PI / 180;
        vec_t r0, v0, a0;
        r0.x = 0;
        r0.y = 0;
        v0.x = 20 * cos(alpha_rad);
        v0.y = 20 * sin(alpha_rad);
        a0.x = 0;
        a0.y = -9.8;

        LabSetColor(LABCOLOR_GREEN);
        DravvAnalyticalPath(&math, &screen, r0, v0, a0, 1);
        DravvAnalyticalPath(&math, &screen, r0, v0, a0, 0.5);

        LabSetColor(LABCOLOR_RED);
        DravvEulerPath(&math, &screen, r0, v0, a0, 1);
        DravvEulerPath(&math, &screen, r0, v0, a0, 0.5);

        LabSetColor(LABCOLOR_YELLOW);
        SimulateEulerPath(&math, &screen, r0, v0, a0);

        LabDrawFlush();
        LabInputKey();
        LabTerm();
    }

    return 0;
}
```

Пример работы программы


Lab Graphics