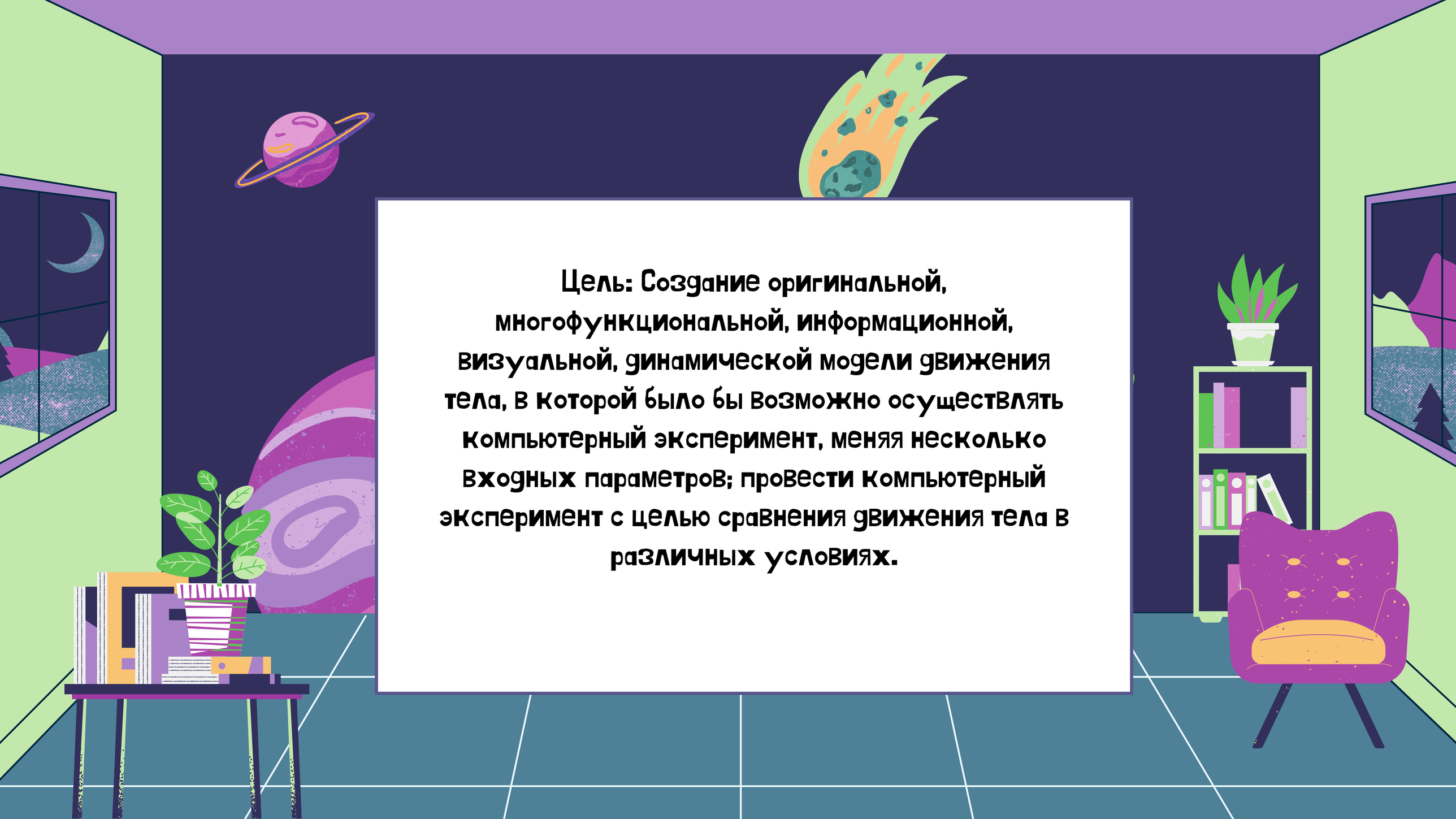
VI Региональный конкурс творческих экспериментальных работ обучающихся Московской области «Архимед»
Номинация "Виртуальное техническое устройство или модель"

Построение и испытание информационных моделей физических процессов

Авторы: Барабанов Андрей, 9 “В” класс,
Петрова Анастасия, 8 “Б” класс
МОУ «Лицей №2 им. В. В. Тихонова» г.о. Павловский Посад Московской области
Научные руководители: Шакурина Мария Борисовна, учитель физики и информатики;
Степанов Сергей Витальевич, учитель физики



Цель: Создание оригинальной, многофункциональной, информационной, визуальной, динамической модели движения тела, в которой было бы возможно осуществлять компьютерный эксперимент, меняя несколько входных параметров; провести компьютерный эксперимент с целью сравнения движения тела в различных условиях.

Задачи, которые мы ставили в начале работы по теме проекта в течение всего времени подготовки, корректировались и изменялись, окончательный перечень выглядит так:

а) подготовить теоретическое обоснование движения тела, брошенного под углом к горизонту;
б) подготовить описательную модель;
в) произвести формализацию;
г) реализовать компьютерную модель;
д) провести компьютерный эксперимент, зафиксировать результаты;
е) спланировать дальнейшее направление работы по теме.

Теоретическое описание движения тела в поле силы тяжести. Построение формальной модели движения тела, брошенного под углом к горизонту.

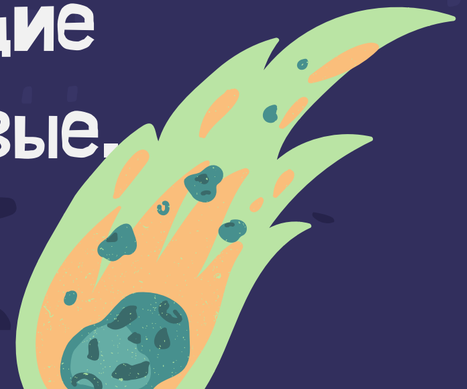
1

Тело, брошенное под углом к горизонту, участвует одновременно в двух движениях: равномерном, прямолинейном по горизонтали, и в равноускоренном по вертикали, в результате движется по параболе. Таким образом, каждое движение происходит по своим законам, независимо друг от друга. Дальность полета определяется произведением проекции начальной скорости на горизонтальную ось на время движения.

2

Высота подъёма тела по вертикальной оси определяется согласно второму уравнению кинематики, где ускорение — это ускорение свободного падения на планете Земля, принятое 9,8 м/с2. Видим, что движение тела, брошенного под углом к горизонту, определяется значением угла наклона и величины начальной скорости движения. Время - третий параметр, принимающий участие в вычислениях, но параметр не регулируемый извне. Время в движении можно разделить на два равных друг другу промежутка, время подъема и время падения.

Стоит обратить внимание на тот факт, что для уравнений кинематики нет границ применимости в виде ограничения - только планета Земля. Закон всемирного тяготения , где действует во всей Вселенной, а сила тяжести как частный случай Закона Всемирного тяготения, существует на любой планете. То есть формулы, описывающие дальность полета и Высоту подъёма одинаковые.



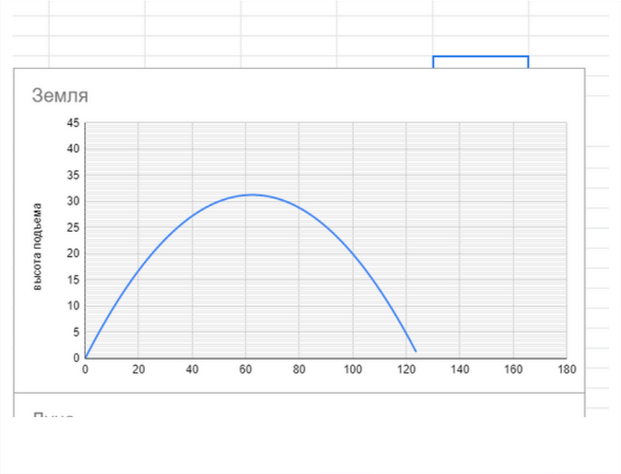
Компьютерная модель движения тела в электронных таблицах

Рисунок 1. Скриншот ЭОР «Баллистическое движение . Движение тела, брошенного под углом к горизонту, издательство «Экзамен-медиа»

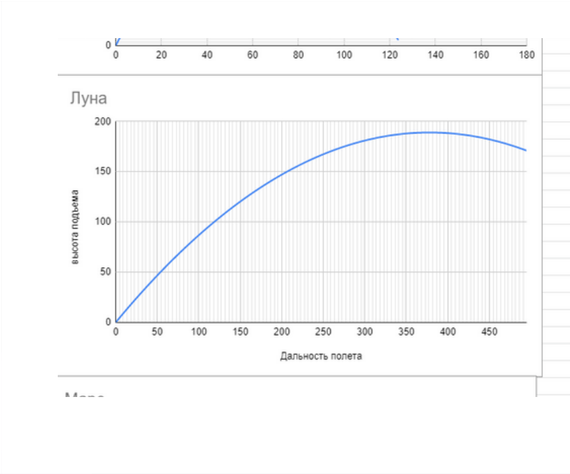
Если вычислять координаты x и y с некоторым шагом времени, то можем построить точки с этими координатами. Получится двумерное изображение траектории движения тела. Сделаем обязательное допущение о том, что исследуемое тело - материальная точка.

В ячейку B5 вводится значение начальной скорости в метрах в секундах, в B6 - значение угла в градусах.
В диапазоне A8:A58 с шагом 0,1 заполнено течение времени.
В ячейку B8 введена формула =$B$4*COS(РАДИАНЫ($B$5))*A8 вычисления дальности полета для Земли, скопирована в диапазон B8:B58.
В ячейку C8 введена формула =$B$4*SIN(РАДИАНЫ($B$5))*A8-4,9*A8*A8 вычисления высоты подъема над поверхностью, затем скопирована в диапазон C8:C58, значение ускорения свободного падения принято 9,8 Н/кг.
Построим график - траекторию движения тела, выбраны диапазоны B8:B58 и C8:C58, тип диаграммы "Гладкий график", настраиваем внешний вид диаграммы.
В ячейки G6, H6, I6, J6 вводим значения ускорения свободного падения на Луне и на других планетах; диапазон ячеек E8:E208 - значение моментов времени с шагом 0,1 секунды.

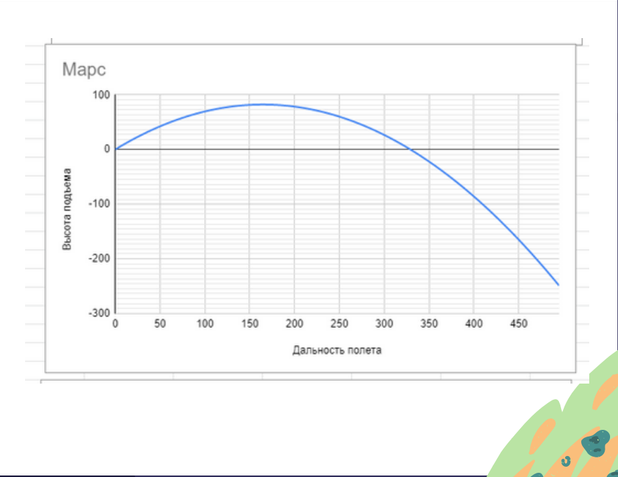
Земля



Луна

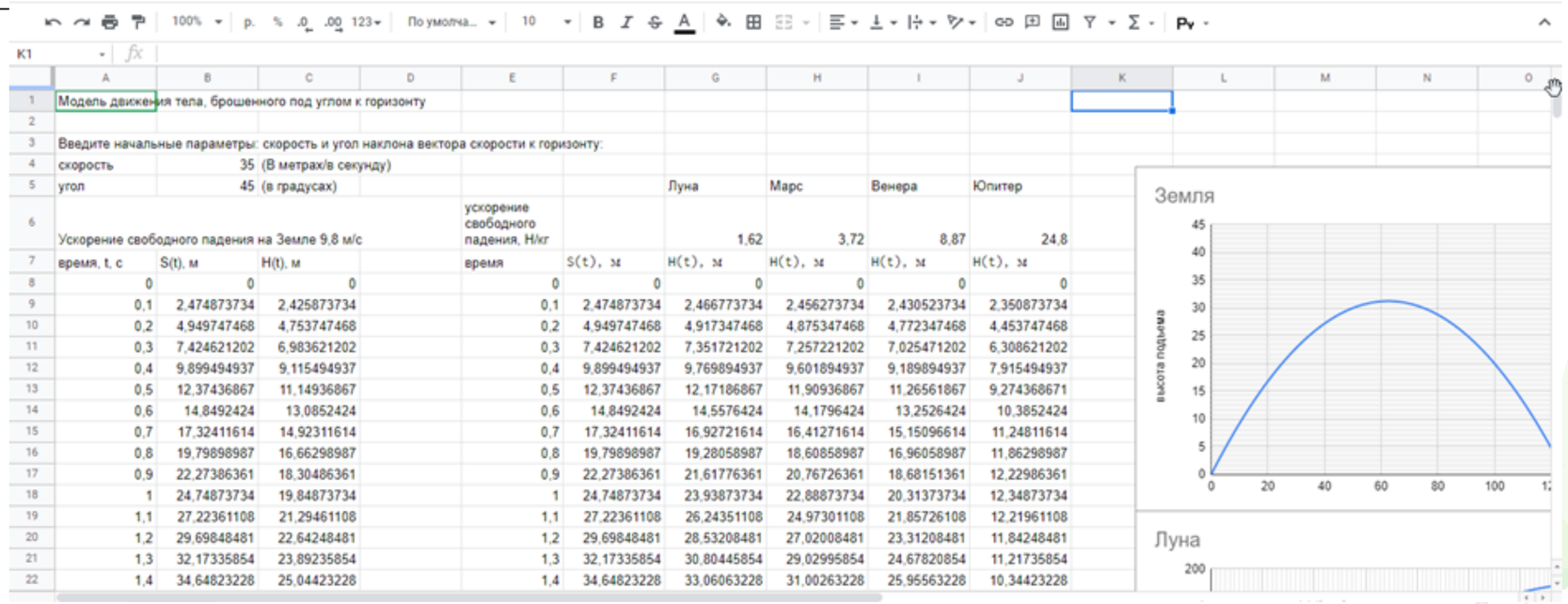


Марс



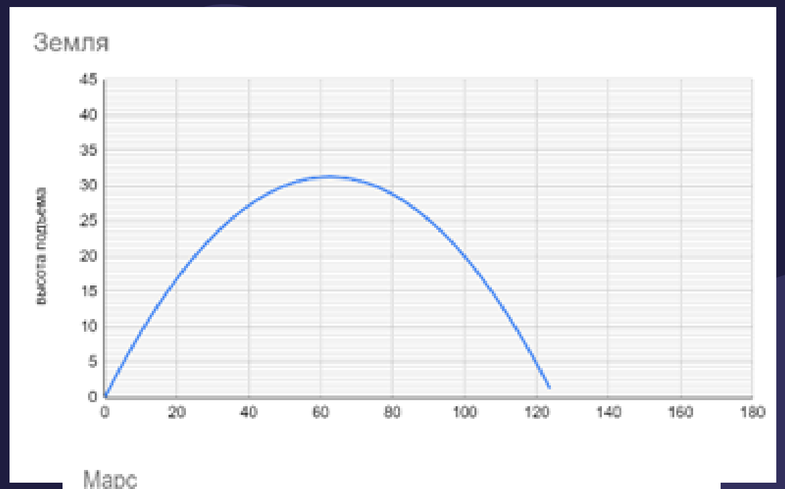
https://docs.google.com/spreadsheets/d/196sqjLln-5zd2Nz2rsyupmIQItiaUt4c3hQa0Wrm9PI/edit?usp=sharing

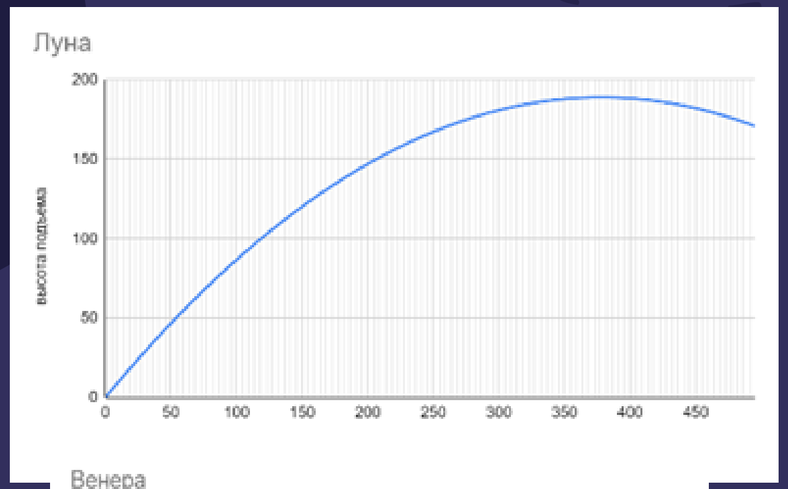
Повторим ввод формулы для определения координат траектории движения тела на других планетах и спутника земли – Луне; заполняем диапазоны ячеек по столбцам до 208 строки. Аналогично настраиваем графики. Модель проверяется на многократных изменениях начальных вводных данных, наблюдаются динамические изменения в заполненных диапазонах, а также на графиках.

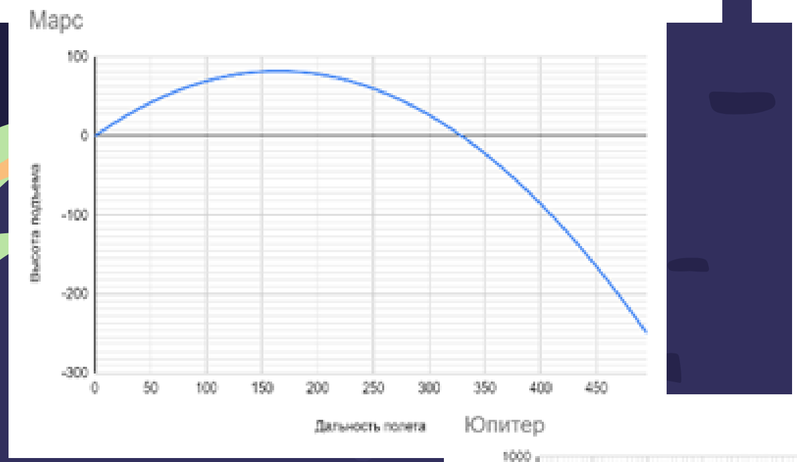


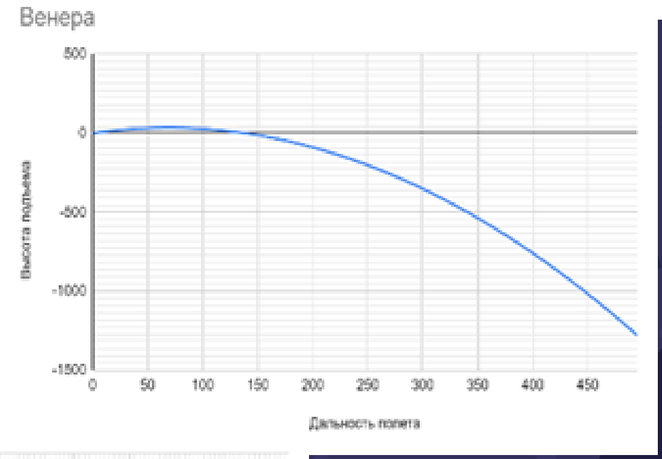
	A	B	C	D	E	F	G	H	I	J
1	Модель движения тела, брошенного под углом к горизонту									
2										
3	Введите начальные параметры: скорость и угол наклона вектора скорости к горизонту:									
4	скорость	35	(В метрах/в секунду)							
5	угол	45	(в градусах)				Луна	Марс	Венера	Юпитер
6	Ускорение свободного падения на Земле 9,8 м/с				ускорение свободного падения, Н/кг		1,62	3,72	8,87	24,8
7	время, t, с	S(t), м	H(t), м		время	S(t), м	H(t), м	H(t), м	H(t), м	H(t), м
8	0	0	0		0	0	0	0	0	0
9	0,1	2,474873734	2,425873734		0,1	2,474873734	2,466773734	2,456273734	2,430523734	2,350873734
10	0,2	4,949747468	4,753747468		0,2	4,949747468	4,917347468	4,875347468	4,772347468	4,453747468
11	0,3	7,424621202	6,983621202		0,3	7,424621202	7,351721202	7,257221202	7,025471202	6,308621202
12	0,4	9,899494937	9,115494937		0,4	9,899494937	9,769894937	9,601894937	9,189894937	7,915494937
13	0,5	12,37436867	11,14936867		0,5	12,37436867	12,17186867	11,90936867	11,26561867	9,274368671
14	0,6	14,8492424	13,0852424		0,6	14,8492424	14,5576424	14,1796424	13,2526424	10,3852424
15	0,7	17,32411614	14,92311614		0,7	17,32411614	16,92721614	16,41271614	15,15096614	11,24811614
16	0,8	19,79898987	16,66298987		0,8	19,79898987	19,28058987	18,60858987	16,96058987	11,86298987
17	0,9	22,27386361	18,30486361		0,9	22,27386361	21,61776361	20,76726361	18,68151361	12,22986361
18	1	24,74873734	19,84873734		1	24,74873734	23,93873734	22,88873734	20,31373734	12,34873734
19	1,1	27,22361108	21,29461108		1,1	27,22361108	26,24351108	24,97301108	21,85726108	12,21961108
20	1,2	29,69848481	22,64248481		1,2	29,69848481	28,53208481	27,02008481	23,31208481	11,84248481
21	1,3	32,17335854	23,89235854		1,3	32,17335854	30,80445854	29,02995854	24,67820854	11,21735854
22	1,4	34,64823228	25,04423228		1,4	34,64823228	33,06063228	31,00263228	25,95563228	10,34423228

Рис.2 Электронная таблица в режиме редактирования









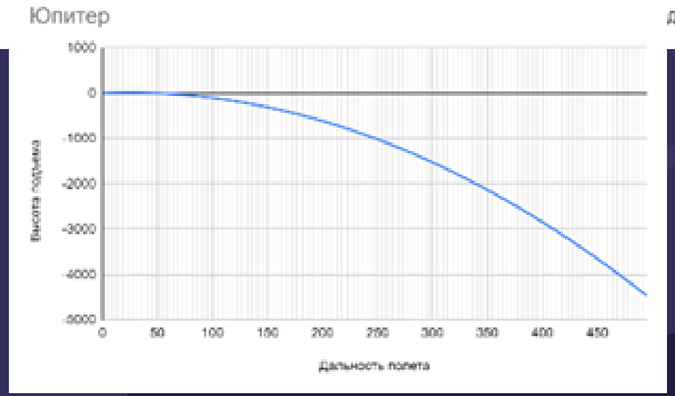


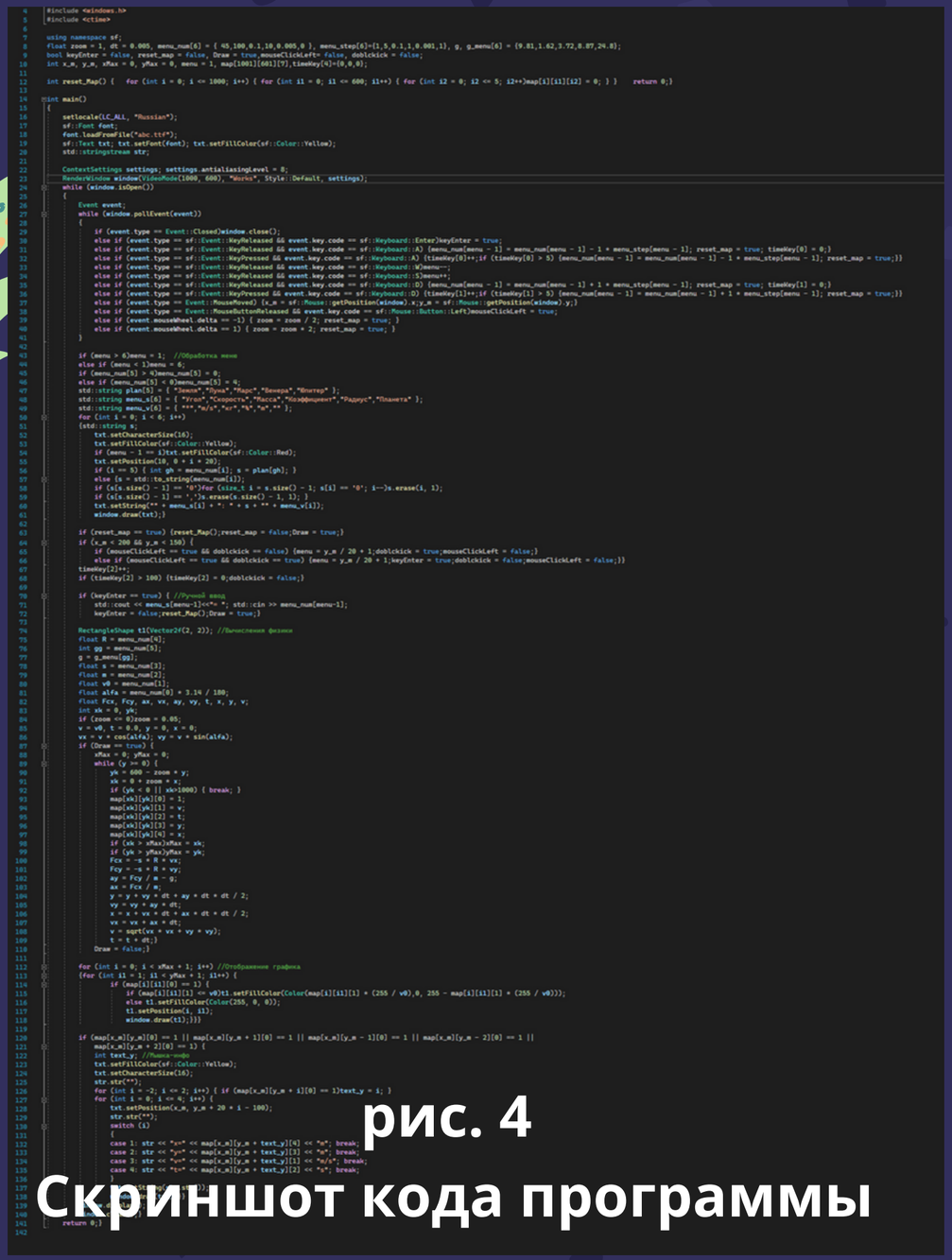
Рис. 3 Графики траектории баллистического движения на разных планетах

Компьютерная модель движения тела на языке C++

В данный момент, программа проходит активную модернизацию и с каждой новой версией, она может измениться почти полностью. В ближайшем будущем, планируется перевод программы на другую формулу, которая сможет повысить точность вычислений.

https://drive.google.com/file/d/1d0mraMCPJ3ujV1WRkSK1ZJKqiRndTa4O/view ссылка для скачивания архива, управление WASD-меню, Q -ручной ввод, скролл колесом мыши – zoom.

Этап разработки оказался самый длительным из всех, создание кода и поиск ошибок заняли большой промежуток времени. Во время создания кода постоянно выявлялись новые проблемы, причиной которых были постоянные модификации его. Из-за частого перебора кода, в нем оставались лишние и неиспользуемые функции и переменные, которые только грузили код, и от которых надо было избавиться. Также не малое внимание было уделено производительности, самая ранняя версия была перегружена циклами, из-за чего график был обновлялся (строился) с частотой, заметной глазу, неточным.



```cpp
#include <windows.h>
#include <ctime>

using namespace sf;
float zoom = 1, dt = 0.005, menu_num[6] = { 45,100,0.1,10,0.005,0 }, menu_step[6]={1,5,0.1,1,0.001,1}, g, g_menu[6] = {9.81,1.62,3.72,8.87,24.8};
bool keyEnter = false, reset_map = false, Draw = true,mouseClickLeft= false, doblckick = false;
int x_m, y_m, xMax = 0, yMax = 0, menu = 1, map[1001][601][7],timeKey[6]={0,0,0};

int reset_Map() {   for (int i = 0; i <= 1000; i++) { for (int i1 = 0; i1 <= 600; i1++) { for (int i2 = 0; i2 <= 5; i2++)map[i][i1][i2] = 0; } }    return 0;}

int main()
{
    setlocale(LC_ALL, "Russian");
    sf::Font font;
    font.loadFromFile("abc.ttf");
    sf::Text txt; txt.setFont(font); txt.setFillColor(sf::Color::Yellow);
    std::stringstream str;

    ContextSettings settings; settings.antialiasingLevel = 8;
    RenderWindow window(VideoMode(1000, 600), "Works", Style::Default, settings);
    while (window.isOpen())
    {
        Event event;
        while (window.pollEvent(event))
        {
            if (event.type == Event::Closed)window.close();
            else if (event.type == sf::Event::KeyReleased && event.key.code == sf::Keyboard::Enter)keyEnter = true;
            else if (event.type == sf::Event::KeyReleased && event.key.code == sf::Keyboard::A) {menu_num[menu - 1] = menu_num[menu - 1] - 1 * menu_step[menu - 1]; reset_map = true; timeKey[0] = 0;}
            else if (event.type == sf::Event::KeyPressed && event.key.code == sf::Keyboard::A) {timeKey[0]++;if (timeKey[0] > 5) {menu_num[menu - 1] = menu_num[menu - 1] - 1 * menu_step[menu - 1]; reset_map = true;}}
            else if (event.type == sf::Event::KeyReleased && event.key.code == sf::Keyboard::W)menu--;
            else if (event.type == sf::Event::KeyReleased && event.key.code == sf::Keyboard::S)menu++;
            else if (event.type == sf::Event::KeyReleased && event.key.code == sf::Keyboard::D) {menu_num[menu - 1] = menu_num[menu - 1] + 1 * menu_step[menu - 1]; reset_map = true; timeKey[1] = 0;}
            else if (event.type == sf::Event::KeyPressed && event.key.code == sf::Keyboard::D) {timeKey[1]++;if (timeKey[1] > 5) {menu_num[menu - 1] = menu_num[menu - 1] + 1 * menu_step[menu - 1]; reset_map = true;}}
            else if (event.type == Event::MouseMoved) {x_m = sf::Mouse::getPosition(window).x;y_m = sf::Mouse::getPosition(window).y;}
            else if (event.type == Event::MouseButtonReleased && event.key.code == sf::Mouse::Button::Left)mouseClickLeft = true;
            else if (event.mouseWheel.delta == -1) { zoom = zoom / 2; reset_map = true; }
            else if (event.mouseWheel.delta == 1) { zoom = zoom * 2; reset_map = true; }
        }

        if (menu > 6)menu = 1;  //Обработка меню
        else if (menu < 1)menu = 6;
        if (menu_num[5] > 4)menu_num[5] = 0;
        else if (menu_num[5] < 0)menu_num[5] = 4;
        std::string plan[5] = { "Земля", "Луна","Марс","Венера","Юпитер" };
        std::string menu_s[6] = { "Угол","Скорость","Масса","Коэффициент","Радиус","Планета" };
        std::string menu_v[6] = { "°","m/s","kr","%","м","" };
        for (int i = 0; i < 6; i++)
        {std::string s;
            txt.setCharacterSize(16);
            txt.setFillColor(sf::Color::Yellow);
            if (menu - 1 == i)txt.setFillColor(sf::Color::Red);
            txt.setPosition(10, 0 + i * 20);
            if (i == 5) { int gh = menu_num[i]; s = plan[gh]; }
            else {s = std::to_string(menu_num[i]);
            if (s[s.size() - 1] == '0')for (size_t i = s.size() - 1; s[i] == '0'; i--)s.erase(i, 1);
            if (s[s.size() - 1] == '.')s.erase(s.size() - 1, 1); }
            txt.setString("" + menu_s[i] + ": " + s + "" + menu_v[i]);
            window.draw(txt);}

        if (reset_map == true) {reset_Map();reset_map = false;Draw = true;}
        if (x_m < 200 && y_m < 150) {
            if (mouseClickLeft == true && doblckick == false) {menu = y_m / 20 + 1;doblckick = true;mouseClickLeft = false;}
            else if (mouseClickLeft == true && doblckick == true) {menu = y_m / 20 + 1,keyEnter = true;doblckick = false;mouseClickLeft = false;}}
        timeKey[2]++;
        if (timeKey[2] > 100) {timeKey[2] = 0;doblckick = false;}

        if (keyEnter == true) { //Ручной ввод
            std::cout << menu_s[menu-1]<<"= "; std::cin >> menu_num[menu-1];
            keyEnter = false;reset_Map();Draw = true;}

        RectangleShape t1(Vector2f(2, 2)); //Вычисления физики
        float R = menu_num[4];
        int gg = menu_num[5];
        g = g_menu[gg];
        float k = menu_num[3];
        float m = menu_num[2];
        float v0 = menu_num[1];
        float alfa = menu_num[0] * 3.14 / 180;
        float Fcx, Fcy, ax, vx, ay, vy, t, x, y, v;
        int xk = 0, yk;
        if (zoom <= 0)zoom = 0.05;
        v = v0, t = 0.0, y = 0, x = 0;
        vx = v * cos(alfa); vy = v * sin(alfa);
        if (Draw == true) {
            xMax = 0; yMax = 0;
            while (y >= 0) {
                yk = 600 - zoom * y;
                xk = 0 + zoom * x;
                if (yk < 0 || xk>1000) { break; }
                map[xk][yk][0] = 1;
                map[xk][yk][1] = v;
                map[xk][yk][2] = t;
                map[xk][yk][3] = y;
                map[xk][yk][4] = x;
                if (xk > xMax)xMax = xk;
                if (yk > yMax)yMax = yk;
                Fcx = -s * R * vx;
                Fcy = -s * R * vy;
                ay = Fcy / m - g;
                ax = Fcx / m;
                y = y + vy * dt + ay * dt * dt / 2;
                vy = vy + ay * dt;
                x = x + vx * dt + ax * dt * dt / 2;
                vx = vx + ax * dt;
                v = sqrt(vx * vx + vy * vy);
                t = t + dt;}
            Draw = false;}

        for (int i = 0; i < xMax + 1; i++) //Отображение графика
        {for (int i1 = 1; i1 < yMax + 1; i1++) {
                if (map[i][i1][0] == 1) {
                    if (map[i][i1][1] <= v0)t1.setFillColor(Color(map[i][i1][1] * (255 / v0),0, 255 - map[i][i1][1] * (255 / v0)));
                    else t1.setFillColor(Color(255, 0, 0));
                    t1.setPosition(i, i1);
                    window.draw(t1);}}}

        if (map[x_m][y_m][0] == 1 || map[x_m][y_m + 1][0] == 1 || map[x_m][y_m - 1][0] == 1 || map[x_m][y_m - 2][0] == 1 ||
            map[x_m][y_m + 2][0] == 1) {
            int text_y; //Поиск-конец
            txt.setFillColor(sf::Color::Yellow);
            txt.setCharacterSize(16);
            str.str("");
            for (int i = -2; i <= 2; i++) { if (map[x_m][y_m + i][0] == 1)text_y = i; }
            for (int i = 0; i <= 4; i++) {
                txt.setPosition(x_m, y_m + 20 + i - 100);
                str.str("");
                switch (i)
                {
                case 1: str << "x=" << map[x_m][y_m + text_y][4] << "m"; break;
                case 2: str << "y=" << map[x_m][y_m + text_y][3] << "m"; break;
                case 3: str << "v=" << map[x_m][y_m + text_y][1] << "m/s"; break;
                case 4: str << "t=" << map[x_m][y_m + text_y][2] << "s"; break;
                }
                txt.setString(str.str());
                window.draw(txt);}}
        window.display();}
    return 0;}
```

рис. 4
Скриншот кода программы

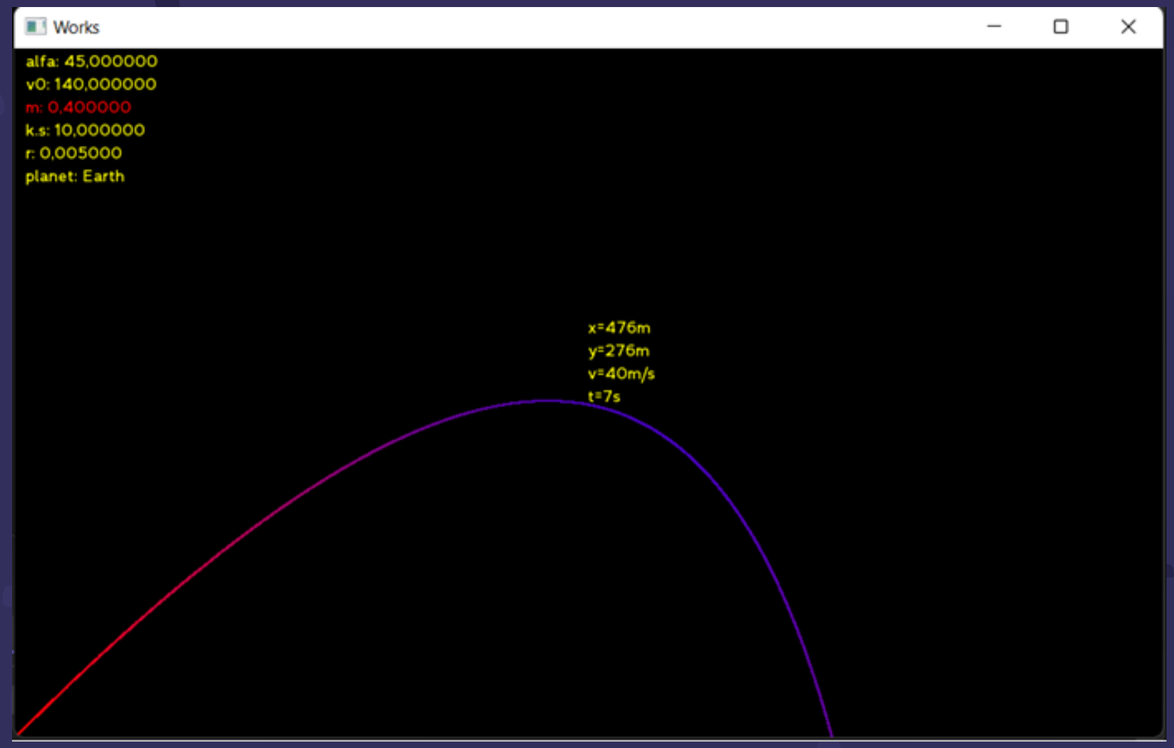


рис. 5
Скриншот работы программы

Перспективы

развития нашего проекты видим в распространении полученных моделей для практического применения не только на уроках физики, астрономии, но и на уроках физической культуры, как теоретический тренажёр в упражнении "метание гранаты"; уже имеются дополнительная литература, в которой рассматривается вопрос физиологии, мышечной силы, скорости реакции; также будет рассматриваться вопрос закона сохранения импульса для увеличения дальности полета тела. Второе возможное направление углубления и расширения исследования: построить модель, учитывающую линейные размеры тела, сопротивление среды. Третье направление, в котором нам хотелось бы продолжить работу над проектом, это создание возможности построения моделей на иных входных условиях, чтобы решались и обратные задачи.

Фотографии или скриншоты соответствующих статей эффективны для закреплении материала урока.

Оставьте максимум места для дополнительных материалов, которыми вы хотите поделиться с учащимися.

Они могут легко вернуться к ним или поискать их в Интернете, если им понадобится освежить знания.

По результатам проведенной работы, можно утверждать, что обе построенные модели соответствуют поставленным целям: они информативны, динамичны, визуальны. Модели доступны для использования, проверены и валидны.

Модели доступны для просмотра и использования по ссылкам:

https://docs.google.com/spreadsheets/d/196sqjLIn-5zd2Nz2rsyupmIQItiaUt4c3hQa0Wrm9PI/edit?usp=sharing

https://drive.google.com/file/d/1d0mraMCPJ3ujV1WRkSK1ZJKqiRndTa4O/view