ГБОУ МО «Одинцовский «Десятый лицей»

**Проект**

**«Симулятор физического эксперимента»**

Проект выполнил:

Бобровников Иван, 10 «Б»

Научный руководитель:

Деткова Людмила Анатольевна

Оглавление

[Паспорт проекта 3](#_Toc130726210)

[Вступление 4](#_Toc130726211)

[Проблема: 4](#_Toc130726212)

[Актуальность: 4](#_Toc130726213)

[Цель: 4](#_Toc130726214)

[Задачи: 4](#_Toc130726215)

[Основная часть 5](#_Toc130726216)

[Теория 5](#_Toc130726217)

[Реализация задачи 1: 5](#_Toc130726218)

[Реализация задачи 2: 6](#_Toc130726219)

[Практика 8](#_Toc130726220)

[Реализация задачи 3: 8](#_Toc130726221)

[Заключение 15](#_Toc130726222)

[Источники информации 15](#_Toc130726223)

[Примечание 16](#_Toc130726224)

# Паспорт проекта

|  |  |
| --- | --- |
| Название | «Симулятор физического эксперимента» |
| Автор | Бобровников Иван Сергеевич, ученик 10 «Б» класса |
| Научный руководитель | Деткова Людмила Анатольевна |
| Цель | Создать веб-приложения для симуляции физического эксперимента. |
| Задачи | 1. Познакомиться с React js. 2. Изучить js библиотеку canvas, предназначенный для создания растрового двухмерного изображения. 3. Создать веб-приложения для симуляции физического эксперимента. |
| Материально-техническое обеспечение | Денежных затрат проект не требует, нужен только компьютер и несколько бесплатных программ |

# Вступление

## Проблема:

Многие ученики, которые сталкиваются с изучением физики, часто испытывают трудности в понимании теории и решении задач. Они считают этот предмет сложным и скучным из-за того, что большинство задач не очень наглядны и трудно представить их визуально. Я же считаю, что физика может быть интересной и увлекательной, если представить ее в более доступном и понятном виде. Поэтому я решил создать инструмент, который поможет ученикам легче усваивать материал и развивать свои навыки в решении физических задач.

## Актуальность:

Этот проект имеет двойную выгоду. Во-первых, он облегчит изучение физики не только мне, но и другим ученикам. Во-вторых, в процессе работы над ним, я смогу познакомиться с новыми технологиями, используемыми для построения графиков, а также улучшить свои навыки в создании сайтов.

## Цель:

В рамках данного проекта планируется разработать веб-приложение, которое будет позволять пользователям проводить симуляцию различных физических экспериментов в интерактивном режиме. Это приложение будет включать в себя набор инструментов для создания и настройки различных физических сценариев, а также возможность визуализации результатов в режиме реального времени.

## Задачи:

1. Познакомиться с React js.
2. Изучить js библиотеку canvas, предназначенный для создания растрового двухмерного изображения.
3. Создать веб-приложения для симуляции физического эксперимента.

# Основная часть

## Теория

### Реализация задачи 1:

React.js — это библиотека для языка программирования JavaScript с открытым исходным кодом для разработки пользовательских интерфейсов. Она помогает быстро и легко реализовать реактивность — явление, когда в ответ на изменение одного элемента меняется все остальное.

React используют для создания одностраничных и многостраничных приложений, разработки крупных сайтов. Например, с его помощью написан стриминговый сервис Netflix и реализованы новостные ленты крупнейших социальных сетей. Библиотека предназначена:

* для создания функциональных интерактивных веб-интерфейсов, работая с которыми, не нужно постоянно обновлять страницу;
* быстрой и удобной реализации отдельных компонентов и страниц целиком — элементы в React легко использовать повторно;
* легкой разработки сложных программных структур — их просто описывать, если использовать реализованный в React подход;
* доработки новой функциональности с любым изначальным стеком технологий: библиотека не зависит от остального инструментария и будет хорошо работать, на чем бы ни был написан код;
* разработки одностраничных и многостраничных приложений (SPA и PWA). Это приложения, которые функционируют как программы и веб-сервисы и имеют соответствующий интерфейс;
* работы с серверной частью сайта или разработки мобильных приложений. В таких случаях React используют совместно с инструментами, адаптирующими веб-технологии под другие цели.

Главной отличительной особенностью React является виртуальное DOM-дерево. DOM-дерево (DOM расшифровывается как Document Object Mode, объектная модель документа) — это древовидная модель, в которой в иерархическом виде собраны все используемые на странице элементы. Особенность React в том, что он создает и хранит в кэше виртуальное DOM-дерево — копию DOM, которая изменяется быстрее, чем реальная структура. Это нужно для того, чтобы быстро обновлять страницы. Если пользователь выполнит действие или наступит какое-либо событие, DOM должна измениться, так как изменятся объекты на странице. Но реальная объектная модель может быть огромной, ее обновление — медленный процесс. Поэтому React работает не с ней, а с виртуальной копией в кэше, которая весит меньше.

Когда происходит событие, из-за которого код должен обновить объект, изменение быстро отображается в виртуальном DOM. После этого обновляется реальная объектная модель. Для пользователя это значит, что изменения на странице отобразятся мгновенно, а не после долгой загрузки.

### Реализация задачи 2:

Canvas (англ. Canvas — «холст», рус. канва́с) — элемент HTML5, предназначенный для создания растрового двухмерного изображения при помощи скриптов, обычно на языке JavaScript. Начало отсчёта блока находится слева сверху.

Основными элементами canvas, которые я буду использовать при создании приложения будут:

* линии. Для рисования линий необходимо несколько элементов:
  1. beginPath() – объявляет начало рисования линий.
  2. strokeStyle – определяет цвет линии.
  3. strokeWidth – определяет ширину линии.
  4. moveTo(x, y) – перемещает курсор в точку с координатами x и y(в пикселях) и делает эту точку началом линии.
  5. lineTo(x, y) – ведет линию до точки с координатами x и y(в пикселях) и делает точку началом следующей линии.
  6. stroke() - проводит линию по предоставленным ранее точкам.
* arc(x, y, radius, startAngle, endAngle, counterclockwise) — рисует дугу с центром в точке с координатами x и y(в пикселях), радиусов radius, от угла startAngle до угла endAngle(от вертикали в направлении вверх), если counterclockwise — false, то дуга будет проведена по часовой и против часовой в противном случае. Если после arc будет идти fill(), то нарисованный круг (или дуга) будет закрашен. Если после arc будет идти stroke(), то круг (или дуга) не будет закрашен.
* fillRect(x, y, width, height) – рисует закрашенный прямоугольник, у которого координаты левого верхнего угла — x и y (в пикселях), высотой height и шириной width.
* strokeRect(x, y, width, height) – рисует закрашенный прямоугольник, у которого координаты левого верхнего угла — x и y (в пикселях), высотой height и шириной width.
* clearRect(x, y, width, height) – очищает прямоугольную область с левым верхним углом в точке с координатами x и y (в пикселях), шириной width и высотой height.
* fillText(text, x, y) – пишет text, начиная в точке с координатам x и y(в пикселях).

## Практика

### Реализация задачи 3:

Я разделю задачу 3 на несколько подзадач:

1. создание интерактивной системы координат и интерфейса веб-приложения
2. добавление движущихся тел
3. добавление системы якорей для движения по окружности и маятников

#### Реализация задачи 3.1:

Для создания системы координат необходимо ввести ряд переменных, которые будут её определять.

* startX и startY будут определять положение начала координат (в пикселях), учитывая что в canvas точка от которой отсчитывается положение объекта находится в левом верхнем углу. Изначально они равны половине ширины и высоты рабочей области соответственно.
* Scale будет определять сколько пикселей будет в каждом делении системы координат.
* numberCof будет определять какое число должно находится на каждом делении системы координат, подстраиваясь при приближении и отдалении.

Все рисование будет происходить в рабочей области шириной width и высотой height. Для того чтобы начать рисовать в элементе <canvas/> необходимо выполнить некоторые приготовления. Для того чтобы указать, где будет происходить рендер содержимого (контекста) необходимо задать этот самый контекст. Делается это с помощью следующего отрезка кода:

const canvasRef = useRef(null);

const canvas = canvasRef.current;

const ctx = canvas.getContext('2d');

После задания контекста все команды для рисования будут иметь вид: ctx.Command, где Command – любая команда canvas.

Теперь можно перейти к отрисовке системы координат. Каждый новый цикл отрисовки надо начинать с очищения рабочий области путем вызова команды ctx.clearRect(0, 0, width, height). После этого надо нарисовать оси системы координат. Для этого проведу две линии:

* первая идет вертикально вниз через точку с координатой startX
* вторая идет горизонтально через точку с координатой startY

Для упрощения понимания положения объекта необходимо выполнить деления и создать сетку. Для этого нужно применить алгоритм в четырех направлениях (вверх, вниз, вправо, влево) от начала координат. Прямую одной из осей координат нужно проводить через каждые scale пикселей, а место пересечения этой прямой отмечать числом. Если ось находится за пределами экрана, то число на оси нужно написать на границе экрана. Чтобы определить, какое число должно быть написано, необходимо вычесть координату данной точки на оси из координаты начала координат, получившееся число нужно разделить на scale и numberCof. Также нужно провести более тонкие линии с интервалами в пять раз меньшие, чем scale, чтобы система координат выглядела более детализированной.

Сейчас я перейду к добавлению интерактивных возможностей в систему координат, начну с функции перемещения. При нажатии кнопки мыши на рабочей области я сохраню положение курсора и при любом движении мыши буду изменять значения startX и startY в соответствии с разницей между текущим и сохраненным положением курсора. Далее я добавил функционал изменения масштаба. Для его реализации я использую прокрутку колесика мыши, при которой плавно увеличиваю или уменьшаю значение scaleCof, а затем вычисляю значение масштаба scale, которое равно с округлением в меньшую сторону при помощи функции floor(). Также я изменяю значение numberCof по формуле .

Интерфейс моего веб-приложения (см. примечание 2) будет состоять из нескольких элементов:

* заголовок в верхней части экрана, в котором будут находиться
* меню в левой части экрана, в котором будет отображаться информация обо всех телах
* рабочая область, заполняющая все оставшуюся часть экрана

#### Реализация задачи 3.2:

Теперь, когда работа с системой координат окончена можно переходить к созданию тел.

Каждое тело будет объектом, обладающем некоторыми параметрами:

* id – уникальный идентификатор для каждого тела
* name – имя каждого тела, не обязательно уникальное
* x и y – начальные координаты тела (в системе координат)
* Vx и Vy – проекции начальной скорости тела на ось x и y системы координат (в метрах на секунду)
* ax и ay – проекции начального ускорения тела на оси x и y системы координат (в метрах на секунду в квадрате)
* isGravity – определяет учитывается ли гравитация при расчете движения тела
* is\_selected – определяет выделено ли тело.
* color – определяет цвет, которым тело будет нарисовано на графике

Всё движение в симуляции зависти от времени, поэтому работу с телами я начал с секундомера. Для того чтобы избежать зависаний секундомер будет считать время с шагом в 0.01 секунды. Управление таймером будет происходить с помощью трех кнопок:

* start – запускает секундомер.
* stop – останавливает секундомер.
* restart – сбрасывает секундомер и останавливает его

Движение тела по каждой из осей можно описать с помощью уравнения: , где position – положение тела в момент времени time, startPosition – начальная координата тела, speed – скорость тела, acs – ускорение тела. Важно, что если на тело действует гравитация, то из acsY необходимо вычесть 10.

Теперь нужно только преобразовать x и y координаты тела в нашей системе координат в координаты canvas для их дальнейшего отображения. Координаты X и Y в системе координат canvas будут вычисляться по формулам: и . Теперь можно спокойно нарисовать тела на график с помощью функции ctx.arc.

Чтобы нарисовать траекторию движения тела, которая не будет зависеть от значения времени нужно для всех координат x видных на экране решить систему уравнений для y, чтобы получить все точки через которые пройде тело. Если в каждой из этих точек нарисовать по точке, то получиться график движения данного тела. Такую же процедуру следует выполнить и в обратную сторону, то есть искать x для каждого y, тогда в графике не будет никаких разрывов. Далее я добавил возможность узнать координату любой точки на траектории. Для этого необходимо сначала определить точку ближайшую к данной на графике. Это делается путем поиска наименьшего значения , где x и y – соответствующие координаты тела, зависящие от времени, mouseX и mouseY – координаты мыши. Если подставить имеющиеся формулы и взять производную, то получиться уравнение, где ax и ay – проекции ускорения на оси координат, Vx и Vy – проекции скорости на оси координат, dx и dy – разности соответствующих координат начального положения тела и координат мыши. R будет минимальным при . Если решить донное уравнение, то можно получить t (время), при котором тело будет максимально близко к положению мыши. Остается только нарисовать точку с соответствующем месте и отметить ее координаты.

Для того чтобы была возможность добавлять тела, я создал в заголовке кнопку create object, после нажатие на которую следующее нажатие в рабочей области графика создаст новое тело. Каждое тело будет отображаться в меню слева в виде трех строк и нескольких кнопок (см. приложение 3):

* первая строка содержит информацию о имени и начальном положении тела
* вторая строка содержит информацию о скорости тела
* третья строка содержит информацию об ускорении тела
* кнопка delete удаляет безвозвратно удаляет тело и всю информацию о нем
* кнопка color открывает меню выбора цвета для данного тела. Я буду использовать библиотеку react-colorful для красивого окна выбора цвета
* кнопка Gravity определяет учитывается ли гравитация при расчете движения тела

#### Реализация задачи 3.3:

Для того чтобы реализовать движение по окружности или маятники нужны неподвижные точки на графике (якоря), к которым можно присоединить тело.

Каждый якорь определяется некоторыми переменными:

* id – уникальный идентификатор якоря
* name – имя якоря
* x и y – координаты якоря на системе координат.

Также необходимо к каждому телу добавить параметр anchore, который будет указывать на id якоря, к которому это тело прикреплено. Это значения равно -1 если тело не прикреплено к якорю. Каждый якорь будет нарисован на системе координат с помощью черного квадрата, от которого идут линии к телам, которые к нему прикреплены. По нажатию на якорь будет появляться меню в левом верхнем углу, в котором можно изменить имя и координату якоря или удалить его (см. приложение 4).

Обработку объектов, прикрепленных к якорю, можно разделить на движение по окружности (без учета гравитации) и колебательное движение (с учетом гравитации).

При движении по окружности тело будет двигаться по круговой траектории с якорем в центре. Однако, само движение немного сложнее. Для расчета движения я буду предполагать, что скорость и ускорение направлены только по касательной к окружности в любой точке траектории. Поэтому, для того чтобы рассчитать движение, необходимо найти проекцию скорости и ускорения на касательную к окружности в точке, где находится тело.

Сначала необходимо определить синус угла между радиус-вектором от якоря до тела и вектором скорости. Для нахождения косинуса угла используется формула , где aX и aY представляют разности соответствующих координат тела и якоря, а bX и bY представляют проекции скорости тела на оси системы координат. Далее, синус угла можно вычислить с помощью основного тригонометрического тождества . Затем проекция скорости на касательную к окружности может быть найдена как , где V представляет скорость тела. Аналогичные действия необходимо выполнить для определения проекции ускорения.

Теперь, когда мы располагаем информацией о скорости и ускорении тела, можем приступать к расчету его движения. Угол, на который тело переместится по окружности, определяется следующей формулой: , где V – скорость тела, acs – ускорение тела, R – радиус окружности. Затем следует определить угол между телом и вертикальной осью на окружности. Для этого можно использовать формулу , где x – координата тела, anchoreX – координата якоря, R – радиус окружности. Сложив эти углы, мы получим угол между телом и вертикальной осью, который обозначим как β. Зная этот угол, мы можем найти координаты тела по формулам ; , где R – радиус окружности, anchoreX и anchoreY – соответствующие координаты якоря.

Далее рассмотрим колебательное движение. Для данного типа движения предполагается, что тело изначально находится в покое и не обладает ни скоростью, ни ускорением. Сначала нужно определить угол между вертикалью вниз и радиус-вектором, идущим от якоря до тела. Для этого можно использовать формулу , где y – координата тела, anchoreY – координата якоря и R – расстояние между телом и якорем. Тогда координату x тела можно описать формулой , где R – расстояние от тела до якоря, anchoreX – координата якоря, g – ускорение свободного падения (9.8 ) и time – текущее время. Чтобы вычислить координату y тела, можно воспользоваться формулой, R – расстояние от тела до якоря и anchoreX – координата якоря.

# Заключение

Я считаю, что проект получился объёмным, и он практически готов. Изучение React js и библиотеки canvas позволило расширить знания о возможностях языка JavaScript и создании интерактивных веб-приложений. Создание веб-приложения для симуляции физического эксперимента дало возможность применить полученные знания на практике и развить умения в решении задач с использованием новых инструментов.

Я убежден, что я создал полезный инструмент, который сможет помочь учащимся изучать физику более эффективно и увлекательно.

Сейчас сайт запускается только на локальной машине, но в будущем я планирую запустить его для общего доступа. Также есть план добавить другие типы симуляций, например: блоки и рычаги; электрические схемы.

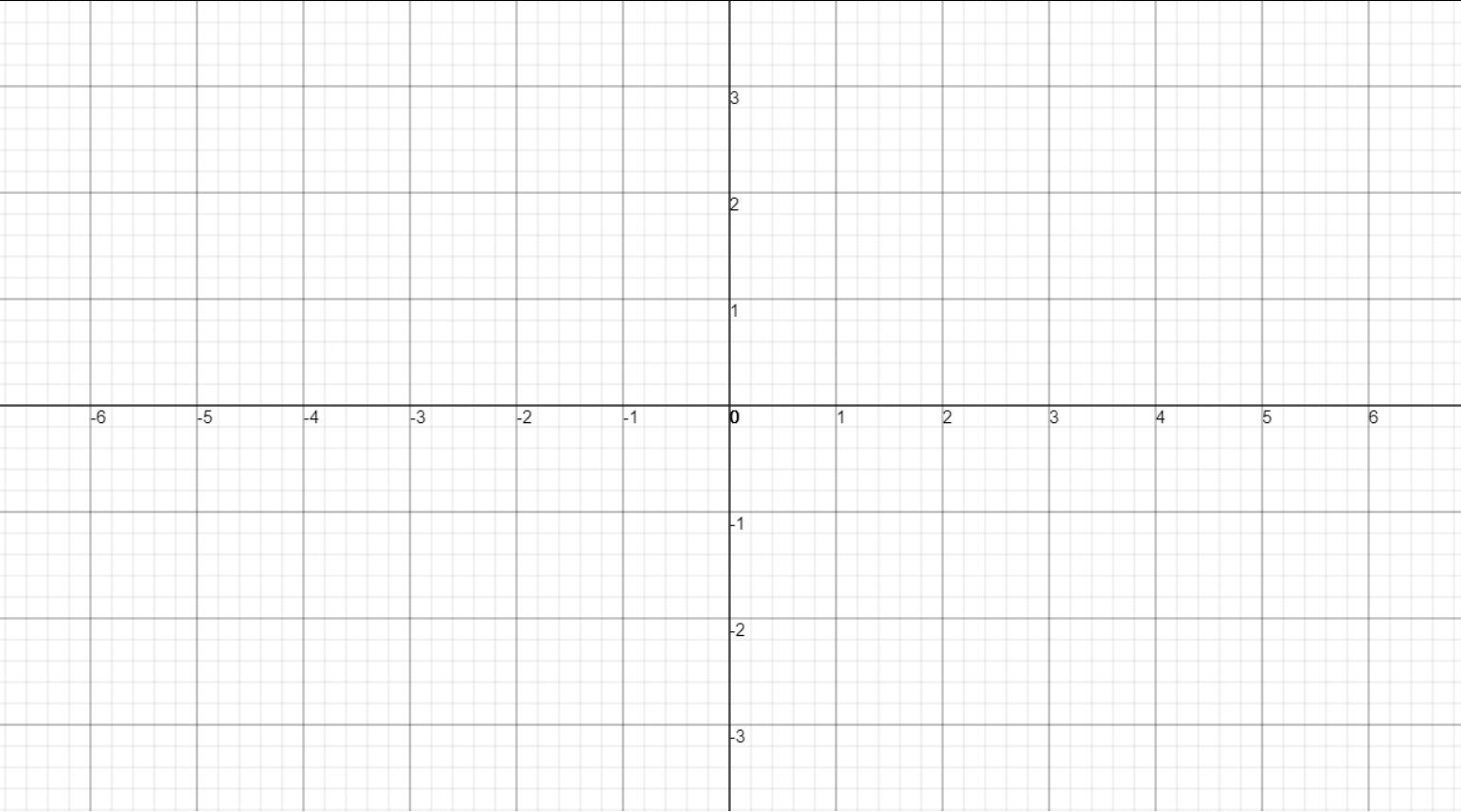
Исходный код проекта с инструкцией по запуску можно найти по ссылке: <https://github.com/IvanBobrovnikov/project.git>

# Источники информации

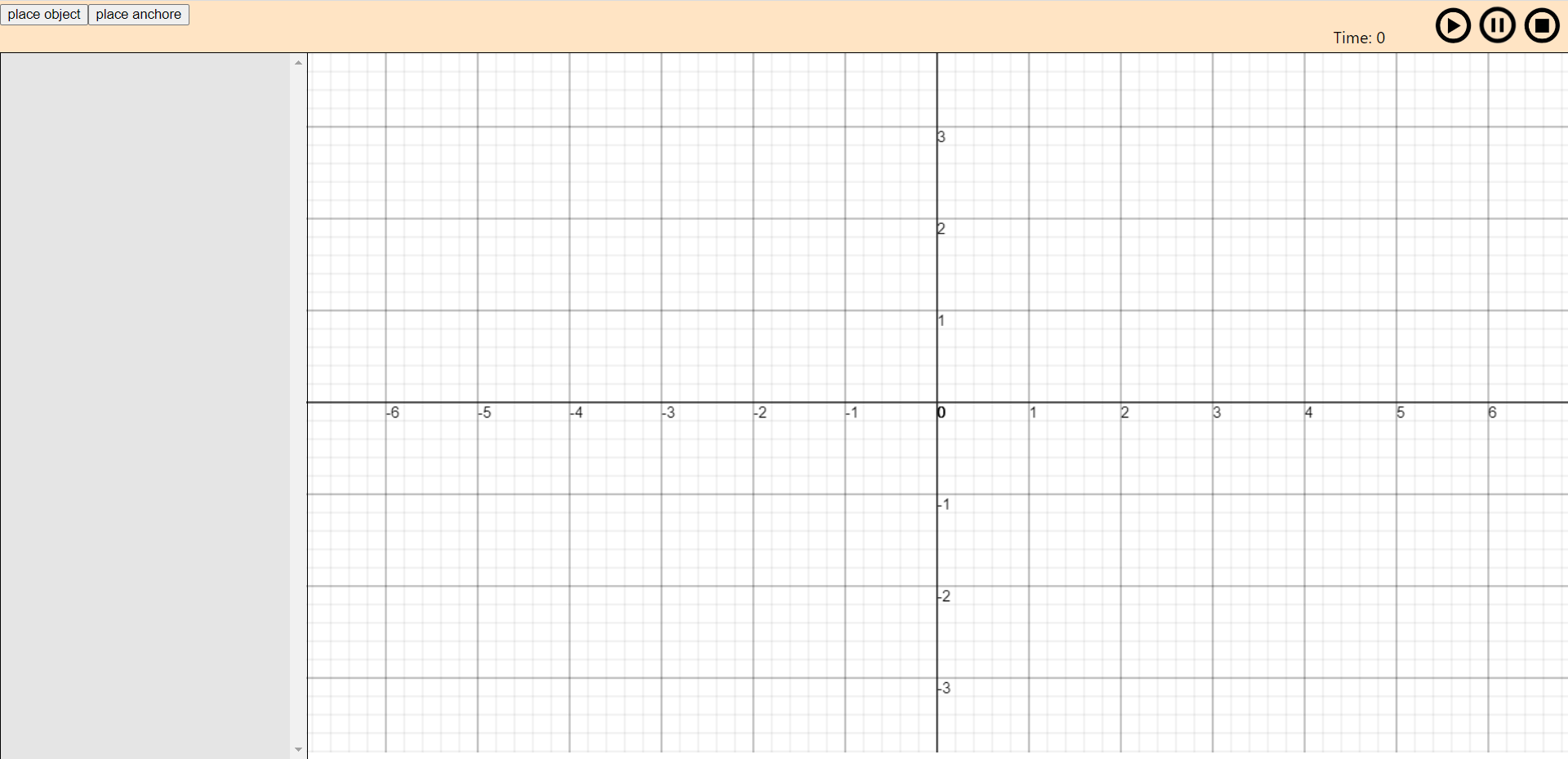
1. Гайд по canvas - <https://www.youtube.com/watch?v=_78uW9yyK7U>
2. Информация о решении кубических уравнений - https://planetcalc.ru/1122/
3. Официальный сайт React js - <https://reactjs.org/>
4. Сайт библиотеки для выбора цвета - <https://www.npmjs.com/package/react-colorful>

# Приложения

1



2



|  |  |
| --- | --- |
| 3 | 4 |

5

