Лабораторная работа №3
«Обработка событий»

Оглавление

Введение 1

Регистрация обработчиков событий 1

События клавиатуры 1

Управляем мячом с клавиатуры 2

- Конструктор `Ball` 2
- Метод `move` 2
- Перевод названий клавиш в команды с помощью объекта 3
- Создаем метод `setDirection` 3
- Реакция на нажатия клавиш 3
- Анимация мяча 3

Запуск программы 4

Задания для самостоятельной работы 5

Обработка события щелчка мышью. 6

Задания для самостоятельной работы (не обязательно) 6

Введение

Мы уже научились работать с холстом, рисовать и раскрашивать объекты, заставлять их двигаться, отскакивать и увеличиваться в размере. Теперь давайте оживим наши программы, добавив в них интерактивности!

Регистрация обработчиков событий

Обработчик события – это асинхронная функция обратного вызова, обрабатывающая события ввода от пользователя, такие как щелчки мышью или нажатия клавиш на клавиатуре. Возможность определения обработчиков позволяет создавать динамические веб-страницы и изменять их содержимое в соответствии с вводом пользователя. Чтобы задействовать обработчик, его нужно зарегистрировать (то есть, сообщить системе, что она должна вызывать функцию-обработчик при появлении указанного события).

События клавиатуры

Состояние клавиатуры можно отслеживать в JavaScript с помощью событий клавиатуры. Каждый раз, когда пользователь нажимает клавишу, генерируется событие. Будем перемещать мяч влево, вправо, вверх или вниз, когда пользователь нажимает на клавиатуре стрелку влево, вправо, вверх или вниз.

Для этого нужно назначить обработчик события `keydown` у объекта `window`, передав ему ссылку на функцию, которая будет вызываться при каждом нажатии клавиши. Таким образом, при

каждом возникновении события `keydown` наш обработчик сможет определить нажатую клавишу и нужным образом на это отреагировать.

Когда возникает событие нажатия клавиши, браузер автоматически вызовет функцию, зарегистрированную в обработчике `keydown` объекта `window`, и передаст в эту функцию параметр – объект события `event`, в свойстве `key` которого содержится информация о том, какая клавиша была нажата.

Управляем мячом с клавиатуры

Напишем программу для управления мячом с клавиатуры. Наша программа будет рисовать мяч и перемещать его вправо. Нажатия клавиш-стрелок будут менять направление мяча, а нажатие на «пробел» остановит его движение. Если мяч уйдет за границу холста, он появится с противоположной стороны. Например, если мяч уйдет за правую границу, он снова появится у левой границы, продолжая двигаться в прежнем направлении, как показано на рис. 1.

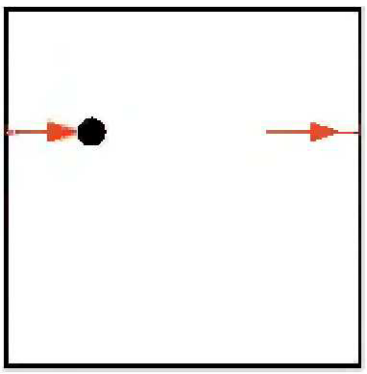

Рис. 1. Если мяч вылетит за правую границу холста, он опять появится слева

Для периодического обновления позиции мяча и перерисовки его в новых координатах мы, как и ранее, воспользуемся функцией `setInterval`.

Конструктор `Ball`

В конструкторе `Ball` мы задали свойствам `x` и `y` (координатам мяча) значения `width/2` и `height/2`, чтобы мяч появился в центре холста. Также мы установили `this.xSpeed` в 5, а `this.ySpeed` в 0 – это значит, что сразу после запуска программы мяч будет двигаться вправо (на каждом шаге анимации его *x*-координата будет увеличиваться на 5 пикселей, а *y*-координата останется неизменной).

Метод `move`

Метод `move`, так же, как и в лабораторной работе №2, обновляет `this.x` и `this.y` на основе значений `this.xSpeed` и `this.ySpeed`. Затем следует код, обрабатывающий выход мяча за границы холста.

Конструкция `if...else` проверяет, не вышел ли мяч за границу холста. Если так и есть, этот код перенесет мяч на противоположную сторону холста. Например, если мяч вышел за левую границу, он должен снова появиться с правой стороны. Иными словами, если `this.x` меньше 0, мы присваиваем `this.x` значение `width`, в результате чего мяч оказывается у правой границы. Последующие строки конструкции `if...else` обрабатывают остальные три границы таким же образом.

Перевод названий клавиш в команды с помощью объекта

Чтобы работать с клавиатурой было проще, создадим объект `keyActions` чтобы определять, какая клавиша нажата, и использовать эту информацию для смены направления полета мяча. Названия полей объекта соответствуют названиям клавиш, а значения – их действиям.

Далее этот объект можно использовать для поиска действия клавиши по ее коду. Например, чтобы узнать действие клавиши пробел, требуется вызвать `keyActions[" "]` и получить строку `"stop"`.

Если среди ключей объекта `keyActions` отсутствует название клавиши, обработчик вернет значение `undefined`.

Создаем метод `setDirection`

Создадим метод `setDirection`, который будет вызываться из обработчика события `keydown`. Обработчик передаст методу `setDirection` информацию о нажатой клавише в виде строки (`"left"`, `"up"`, `"right"`, `"down"` или `"stop"`). На основе этого `setDirection` будет менять свойства мяча `xSpeed` и `ySpeed` таким образом, чтобы направление полета мяча соответствовало направлению, выбранному нажатием клавиши. Например, если методу передана строка `"down"`, мы сбросим `this.xSpeed` в 0, а `this.ySpeed` присвоим значение 5.

Тело этого метода представляет собой одну большую конструкцию `if...else`. Новое направление передается в аргументе `direction`: если это `"up"`, то сбросим свойство `xSpeed` в 0, а `ySpeed` присвоим значение -5. Остальные направления обрабатываются аналогично.

Наконец, если в метод передана строка `"stop"`, мы сбросим в 0 оба свойства `xSpeed` и `ySpeed`, что приведет к остановке мяча.

Реакция на нажатия клавиш

Создадим объект-мяча с помощью конструктора `Ball`.

Будем отслеживать события `keydown`, чтобы менять направление мяча в соответствии с нажатыми клавишами.

Воспользуемся объектом под названием `keyActions`, чтобы определять, какая клавиша нажата, и использовать эту информацию для смены направления полета мяча.

Таким образом, в `direction` оказывается направление: `"left"`, если нажата стрелка влево, `"right"` для стрелки вправо, `"up"` для стрелки вверх, `"down"` для стрелки вниз и `"stop"` для пробела. Если была нажата какая-то другая клавиша, `direction` примет значение `undefined` и на анимацию это никак не повлияет.

Для осуществления смены направления полета мяча требуется вызвать метод `setDirection` у объекта класса `Ball`. Когда возникает событие нажатия клавиши, браузер автоматически вызывает функцию `divertDirection`, зарегистрированную для отслеживания события `keydown` с единственным предопределенным параметром – объектом события, содержащем информацию о нажатой клавише. Поэтому для доступа к локальному объекту `ball` внутри функции `divertDirection`, последнюю нужно определить внутри функции `main`. Такой прием позволяет избежать необходимости использовать глобальные переменные.

Анимация мяча

Далее остается только анимировать мяч. Аналогично предыдущим анимациям, для периодического обновления позиции мяча будем использовать метод `setInterval`.

Мы используем `setInterval`, чтобы вызывать функцию анимации раз в 30 миллисекунд. Функция сначала очищает весь холст с помощью `clearRect`, а затем вызывает методы объекта-мяча `draw` и `move`. Как мы уже знаем, метод `draw` просто рисует окружность в текущей позиции мяча, а метод `move` обновляет позицию мяча в соответствии со значениями его свойств `xSpeed` и `ySpeed`.

Запуск программы

При запуске вы увидите черный мяч, движущийся слева направо по холсту, как на рис. 2. Дойдя до правого края, мяч должен появиться с левой стороны, продолжая движение вправо. При нажатии клавиш-стрелок мяч должен менять направление, а при нажатии на «пробел» остановиться.

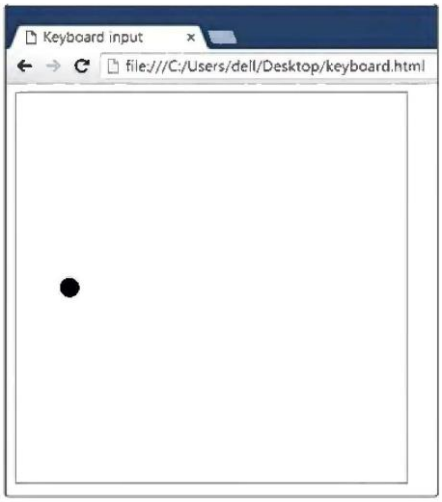

Рис. 2. Скриншот анимации мяча

Если анимация не реагирует на нажатия клавиш, кликните по странице, чтобы JavaScript мог отслеживать нажатия клавиш.

Задания для самостоятельной работы

- Отскоки от границ холста

  Измените код так, чтобы мяч отскакивал от границ при столкновении, а не появлялся с другой стороны.

  Подсказка: при столкновении просто измените направление на противоположное.

- Управление скоростью

  Сейчас на каждом шаге анимации мяч перемещается на 5 пикселей, поскольку `setDirection` всегда задает свойствам `xSpeed` и `ySpeed` значения -5 или 5. Создайте в конструкторе `Ball` новое свойство `speed`, хранящее скорость мяча, и используйте его в методе `setDirection` вместо цифры 5.

  Теперь добавьте в код возможность задавать скорость (`speed`) нажатием цифровых клавиш от 1 до 9.

  Подсказка: создайте объект `speeds` с разными значениями скоростей и в обработчике события `keydown` выбирайте из него соответствующую клавише скорость.

- Гибкое управление

  Измените код так, чтобы при нажатии клавиши Z мяч замедлял движение, а при нажатии X разгонялся. Затем сделайте так, чтобы клавиша C уменьшала размер мяча, а V – увеличивала.

  Что произойдет, если скорость станет отрицательной? А размер?

  Добавьте в код проверку, гарантирующую, что скорость и размер никогда не станут меньше 0.

Обработка события щелчка мышью.

Событие `click` элемента `<canvas>` позволяет отслеживать щелчки мыши в этом элементе. Координаты указателя мыши в момент щелчка хранятся в объекте события, который передается браузером в аргументе `event`. Извлечь координаты из объекта `event` можно обратившись к свойствам `event.clientX` и `event.clientY`. Однако эти координаты нельзя использовать непосредственно, поскольку они соответствуют положению указателя мыши в клиентской области окна браузера, а не в элементе `<canvas>` (см. рис. 3).

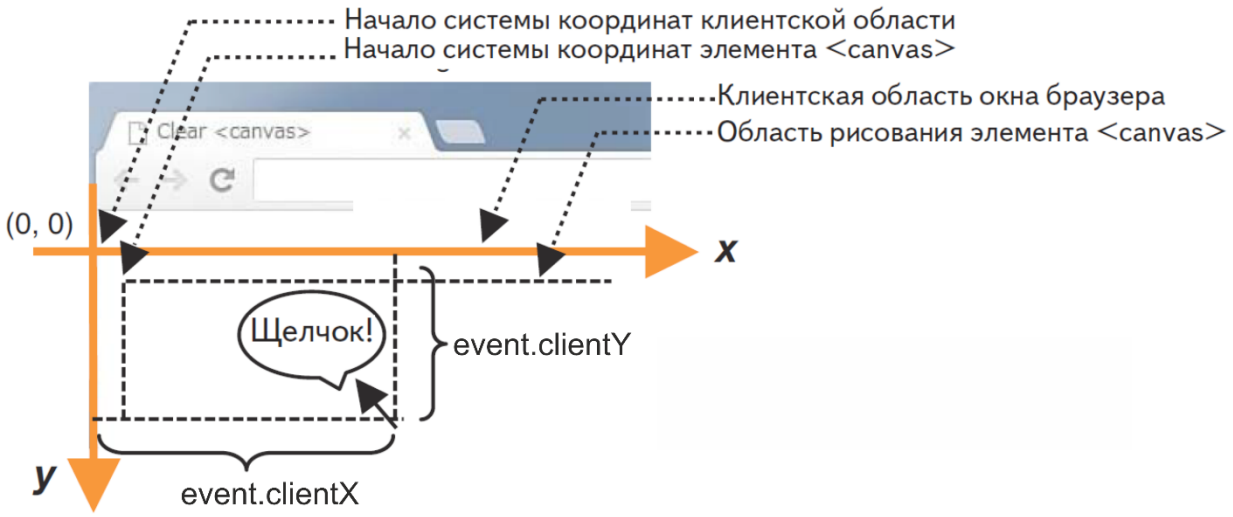

Рис. 3. Система координат клиентской области окна браузера и координаты элемента `<canvas>`

Следовательно, нужно преобразовать координаты из системы координат клиентской области окна браузера в систему координат элемента `<canvas>`.

Это можно сделать с помощью координат верхнего левого угла `rect.left` и `rect.top` элемента `<canvas>` в клиентской области окна браузера (см. рис. 3), который возвращается с помощью следующей команды:

```
const rect = canvas.getBoundingClientRect();
```

То есть, выражение (`x - rect.left`) и выражение (`y - rect.top`) смещают начало координат в позицию верхнего левого угла элемента `<canvas>`.

Задания для самостоятельной работы (не обязательно)

Определите функцию-обработчик `onclick()`, которая выполняет следующие действия:

1. Получает координаты указателя мыши в момент щелчка.

2. Рисует в этом месте точку.