Бесконечный раннер

Из «Словаря разработчика» — вашего проводника в мир игр

Это запись о жанре «бесконечный раннер». Об аналогичных играх см. статью «Платформеры».

Популярность бесконечных раннеров возросла с появлением смартфонов и планшетов. Хотя сама концепция использовалась в игровых автоматах еще в 1980-х, первой широко признанной игрой стал выпущенный в 2009 году раннер Canabalt для смартфонов.

Бесконечный раннер

Жанр:	бесконечный раннер
Режим:	для одного игрока
Первый выпуск:	Canabalt (2009)
Время игры:	от 2 минут (зависит от уровня)
Требуемые навыки:	быстрая реакция, наблюдательность

В бесконечных раннерах обычно используется всего одна кнопка, заставляющая персонаж прыгать. Движение возможно только вперед, кроме того, игрок не может контролировать свою скорость. Нужно оставаться живым как можно дольше, уклоняясь от препятствий и избегая попадания в ловушки. По мере продвижения по уровням скорость перемещения увеличивается. Игра завершается в момент смерти персонажа.

Бесконечные раннеры замечательно подходят для смартфонов и планшетов, ведь для управления персонажем достаточно в нужный момент касаться пальцем экрана. Это дает возможность играть где угодно. А желание достичь более высоких уровней и получить больше очков делает игру по-настоящему захватывающей.

Из знаменитых бесконечных раннеров для смартфонов и планшетов следует упомянуть Temple Run, Jetpack Joyride и Robot Unicorn Attack. Игра Temple Run насчитывает более 100 миллионов скачиваний. Элементы этого жанра используются и в других играх.

БЕСКОНЕЧНЫЙ РАННЕР

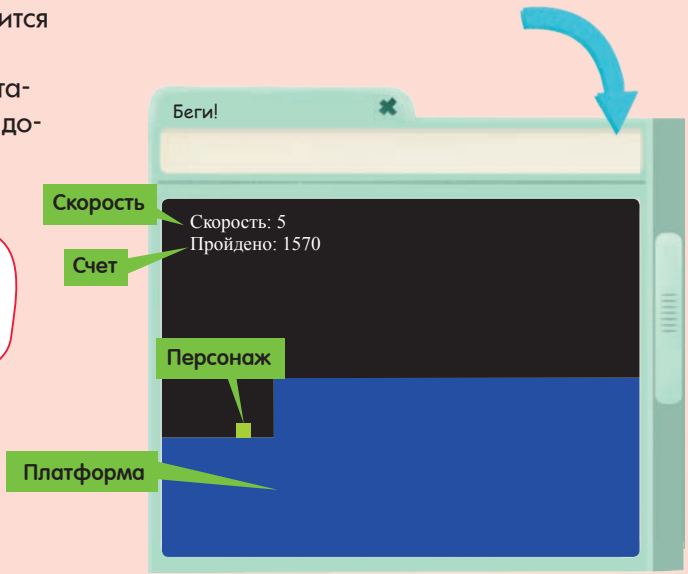
Бесконечные раннеры представляют собой разновидность платформеров. В нашей версии будет всего один элемент управления, отвечающий за прыжки. Персонажу нужно как можно дольше оставаться в живых. Если он не успеет запрыгнуть на платформу и врежется в нее, он умрет.

Отличие бесконечного раннера от других игр в том, что игровой мир в них создается процедурной генерацией. То есть мы не будем программировать уровни вручную, как в «Змейке» и «Настольном теннисе», а напишем код, который будет автоматически генерировать новые уровни в процессе игры.

Создание игры

Правила нашего бесконечного раннера очень просты. Персонаж бежит, запрыгивая на появляющиеся перед ним платформы. Нужно вовремя совершать прыжки, чтобы не врезаться в край платформы, иначе игра завершится. Чем дольше продолжается игра, тем быстрее бежит персонаж. А значит, вовремя запрыгивать на платформы становится сложнее.

В нашей версии игры для прыжка достаточно будет нажать любую клавишу. Мы добавим силу тяжести и распознавание столкновений. Кроме того, мы напишем код генерации пола, причем размер платформ должен все время меняться. Мы будем следить за скоростью персонажа и пройденным им расстоянием, давая игроку возможность превзойти собственные более ранние результаты. Игра будет иметь вот такой вид:



ПЕРВАЯ ПЛИТКА ПОЛА

Персонаж нашего бесконечного раннера будет прыгать по платформам. Платформы, формирующие пол, создаются из плиток разной высоты.
Так как игровой мир будет сделан процедурной генерацией, то есть уровни в процессе игры сформирует сам код, — мы начнем с написания кода первой плитки.

В некоторых играх для создания мира применяются небольшие графические фрагменты, называемые ПЛИТКАМИ.

Создание пола

В «Настольном теннисе» мы, благодаря функциям, получили объекты `player` и `AI`. Создать плитки пола поможет функция `floor`:

```
function floor(x, height) {
    this.x = x;
    this.width = 700;
    this.height = height;
}
```

Функция floor
Левый угол плитки
Ширина плитки
Высота плитки

Функция `floor` принимает два аргумента: X и `height`. Положение верхнего левого угла плитки даст нам X-координату. Ширина плитки составит 700 пикселов. Высота в процессе игры будет меняться, чтобы дать персонажу возможность прыгать с одной плитки на другую.

Мир игры

Игровой мир мы создадим как объект `world` с набором свойств `width` и `height` элемента `<canvas>`, значения которых будут `640` и `480` пикселов. Свойство `gravity`, нужное в дальнейшем, составит 10 пикселов на тик. Первую плитку пола мы создадим в массиве `floorTiles`.

```
var world = {
    height: 480,
    width: 640,
    gravity: 10,
    floorTiles: [
        new floor(0, 140)
    ]
};
```

Объект world
Высота
Ширина
Сила тяжести
Плитка пола
Массив
Новый оператор

Внутри массива `floorTiles` вызовем только что определенную функцию `floor` со значениями 0 и 140 в качестве аргументов. Оператором `new` создадим первую плитку пола в левом углу холста. Ее X-координата будет равна 0, а высота составит 140 пикселов.

Цикл для рисования плиток пола

В процессе игры должны непрерывно генерироваться плитки. Для этого в массиве `floorTiles` используем цикл for. Он будет находиться внутри объекта `world` и являться частью функции `draw`, рисующей каждую плитку на элементе `<canvas>`:

```
draw: function() {
     ctx.fillStyle = "black";
     ctx.fillRect(0, 0, this.width, this.height);
     for(index in this.floorTiles) {
          var tile = this.floorTiles[index];
          var y = world.height - tile.height;
          ctx.fillStyle = "blue";
          ctx.fillRect(tile.x, y, tile.width, tile.height);
     }
}
```

Массив плиток пола

Цикл for

Оператор вычитания

Для рисования плитки нужно вычислить ее Y-координату (высоту). Для этого вычтем свойство `tile.height` из значения `world.height`, указывающего на нижнюю часть элемента `<canvas>`. Полученное число скажет нам, плитку какой высоты нужно нарисовать. Оно будет меняться, так как нам требуются плитки разной высоты. Цикл должен иметь возможность доступа к разным значениям в массиве `floorTiles`. Для этого с помощью индекса и ключевого слова `in` организуем отсчет вверх, начиная с нуля.

Сделаем игровой мир черным, а плитки синими. Используем контекст нашего элемента `<canvas>`, как в предыдущих заданиях. Для заливки первой плитки передадим методу `fillRect` внутри цикла X- и Y-координаты плитки, а также параметры `tile.width` и `tile.height`.