Как создать свой набор задач для Робота-Blockly?

Что это такое?

Робот-Blockly [1] – это приложение для обучения программированию, в котором исполнитель Робот из системы «Исполнители» [2] управляется с помощью программы, собранной из блоков (как в среде Scratch).

На странице [1] (см. список ссылок ниже) размещены несколько готовых наборов задач, которые можно использовать как в режиме онлайн, так и локально (на компьютере, не имеющем доступа к Интернету).

Оффлайн-версию можно скачать со страницы [1] . Эта версия позволит вам создать собственные наборы задач для Робота, собрать собственную палитру блоков и установить собственные ограничения на количество используемых блоков. Про то, как это сделать, рассказывает этот документ.

Если у вас получился удачный набор задач и вы готовы поделиться им с коллегами, присылайте архив с файлами на почту kpolyakov@mail.ru.

Ссылки:

[1] Робот-Blockly: http://kpolyakov.spb.ru/school/robots/blockly.htm

[2] Система «Исполнители»: http://kpolyakov.spb.ru/school/robots/robots.htm

1. Скачивание архива

Сначала нужно скачать архив `rblockly.zip` со страницы [1] или по прямой ссылке:

http://kpolyakov.spb.ru/loadstat.php?f=/download/rblockly.zip

Распакуйте архив в отдельный каталог. Вы должны увидеть каталоги

`js` – скрипты, которые используются при работе приложения;

`media` – рисунки и звуковые файлы;

`solutions` – решения всех задач, опубликованных на странице [1].

2. Создание файлов вашего приложения

Для создания своего приложения необходимо отредактировать два файла – веб-страницу на языке HTML и файл на языке JavaScript, в котором описываются все уровни игры. Рекомендуется давать этим файлам имена с одинаковым окончанием. Например, пусть мы хотим создать блок задач по искусственному интеллекту (AI – artificial intelligence). Тогда можно выбрать окончание ai, так что наши файлы будут называться

`robot_ai.html`

`robot_ai.js`

Файл `robot_ai.html` должен быть в корне, то есть в том же каталоге, где находятся все HTML-файлы. А файл `robot_ai.js` нужно записать в подкаталог `js`.

Для упрощения жизни в архиве есть файлы-заготовки, с которых можно начать работу:

`robot_.html`

`js/robot_.js`

Создайте их копии с именами `robot_ai.html` и `robot_ai.js` (важно, чтобы второй файл был создан в подкаталоге `js`).

Редактировать эти файлы можно в любом текстовом редакторе, например, в Блокноте, NotePad++, Sublime Text и др.

3. Редактирование HTML-страницы

1) Откройте файл `robot_ai.html`. Найдите в начале файла строки

```
<!-- Измените название вашего приложения  -->
<title>Робот: введите здесь название набора задач</title>
<!-- -->
```

Вместо текста, выделенного маркером, введите название своего приложения. Эта строка появится в заголовке страницы и в самом верху тела страницы.

2) Затем найдите в том же файле строчки

```
<!-- Добавьте ссылку на файл, который содержит набор задач  -->
<script src="js/robot_.js"></script>
<!-- -->
```

и исправьте название JavaScript-файла на `robot_ai.js`:

```
<!-- Добавьте ссылку на файл, который содержит набор задач  -->
<script src="js/robot_ai.js"></script>
<!-- -->
```

4. Редактирование полей для Робота

Все остальные данные хранятся в файле `robot_ai.js`.

Найдите в начале файла `robot_ai.js` массив `Maps`, начало которого выглядит примерно `так:`

```
var Maps =
 [ [], // Level 0
  ['.........',  // Level 1
    '.........',
    '.........',
    '.........',
    '.........',
    '.........',
    '.........',
    '.........',
    '.........',
    '.........',
```

```
    '.........' ],
  ['.........',  // Level 2
   '.........',
   '.........',
   '.........',
   '.........',
```

и т.д.

Это массив массивов, в котором каждый уровень описывается как массив символьных строк. Поле Робота представляет собой массив размером 10 на 10 клеток, поэтому карта уровня – это массив из 10 строк, каждая из которых состоит из 10 символов. Можно использовать и другие размеры поля, но при этом графика будет некачественной.

Уровень 0 не используется. В листинге (см. выше) жёлтым маркером выделена карта уровня 1.

Точки означают, что пока все клетки поля пустые. На карте поля можно использовать следующие символы:

`w` – стена;

`x` – грядка, где нужно посадить цветы;

`p` – клумба с цветами;

`b` – база, куда Робот должен придти для подзарядки после выполнения задания.

Робот на поле обозначается цифрой, каждая цифра обозначает определённое положение Робота:

0 – Робот смотрит вверх («на север»);

2 – Робот смотрит направо («на восток»);

4 – Робот смотрит вниз («на юг»);

6 – Робот смотрит налево («на запад»).

Можно выбрать начальное положение Робота на грядке, в этом случае его нужно обозначить заглавной латинской буквой **X**. При этом Робот смотрит вверх («на север»). Можно использовать также буквы

Y – Робот на грядке, смотрит вправо («на восток»)

Z – Робот на грядке, смотрит вниз(«на юг»)

W – Робот на грядке, смотрит влево («на запад»)

Робота можно расположить и на Базе, в этом случае его нужно обозначить заглавной латинской буквой **B**. При этом Робот смотрит вверх («на север»). Можно использовать также буквы

C – Робот на базе, смотрит вправо («на восток»)

B – Робот на базе, смотрит вниз(«на юг»)

У – Робот на базе, смотрит влево («на запад»)

Приведём для примера карту одного из уровней блока задач по линейным алгоритмам:

```
['.........',  // Level 6
 '.........',
 '....b.....',
 '...w.ww...',
 '...wxxw...',
 '...wwww...',
 '....0.....',
 '.........',
 '.........',
 '.........' ],
```

Все строки одного уровня должны быть одинаковой длины. На карте нельзя использовать неизвестные символы.

5. Проверка решения на нескольких обстановках для Робота

Для того чтобы проверка решения проходила на нескольких обстановках, вместо одной карты (массива строк) для данного уровня нужно записать массив карт (массив из массивов строк). Например, так:

```
var Maps =
 [ [], // Level 0 not used
//--- Level 1 ------------------------------------
 [ // начало массива карт
  ['.........',  // Level 1-1 – первая карта
   '.........',
   '.........',
   '.........',
   '....b.....',
   '.....0w...',
   '.........',
   '.........',
   '.........',
   '.........' ],
  ['.........',  // Level 1-2 – вторая карта
   '.........',
   '.........',
   '.........',
   '.........',
```

```
        '....b0....',
        '..........',
        '..........',
        '..........',
        '..........' ],
      ['..........',  // Level 1-3 – третья карта
        '..........',
        '..........',
        '...w..w...',
        '...wb.w...',
        '...w.0w...',
        '...w..w...',
        '..........',
        '..........',
        '..........' ],
  ], // конец массива карт для уровня 1
  ...
```

6. Редактирование ограничений

Для каждого уровня вы можете задать максимальное количество блоков, которые можно использовать в решении. Найдите в файле `robot_ai.js` массив `BlockLimit`:

```
var BlockLimit = [0, // Level 0 unused
    50,  // Level 1
    50,  // Level 2
    50,  // Level 3
    50,  // Level 4
    50,  // Level 5
    [30, 40, 50],  // Level 6
    [30, 40, 50],  // Level 7
    [30, 40, 50],  // Level 8
    [30, 40, 50],  // Level 9
    [30, 40, 50],  // Level 10
     ];
```

Предельное количество блоков задаётся для каждого уровня отдельно. Сейчас на первых пяти уровнях установлено ограничение в 50 блоков. Для следующих уровней установлено три ступени ограничений (массив). Наибольшее число в массиве (50) – это максимальное число блоков, которые ученик может использовать в программе. При этом решение засчитывается и решение получает рейтинг «три звезды». Следующее число

(40) – это максимальное число блоков, за которое можно получить «серебряный кубок» и рейтинг «четыре звезды». Первое число означает количество блоков, за которое программа получает рейтинг «пять звёзд» и «золотой кубок».

Если вы изменяете количество уровней, проверьте, что количество значений в массиве `BlockLimit` вы тоже изменили, иначе будет ошибка.

Вы можете задать предельное количество блоков какого-то типа. Например, вы хотите, чтобы ученик на уровне 5 использовать блок `посади` только два раза, а цикл `повторить N раз` – два раза. Найдите в файле `robot_ai.js` массив `someBlockLimit` и добавьте ограничения к уровню `Level 5`:

```
var someBlocksLimit = [ {}, // Level 0 unused
 {  }, // Level 1
 {  }, // Level 2
 {  }, // Level 3
 {  }, // Level 4
 { 'robot_plant': 2, ' controls_repeat_ext': 1 }, // Level 5
 {  }, // Level 6
 {  }, // Level 7
 {  }, // Level 8
 {  }, // Level 9
 {  }, // Level 10
];
```

В этом массиве ограничения каждого уровня задаются в виде объекта JavaScript (словаря). Ключи этого словаря – это названия блоков, а значения – наибольшее разрешенное количество блоков этого типа. Как только это количество блоков израсходовано, блок в палитре становится неактивным.

Приведём кодовые названия некоторых блоков:

вперёд	robot_forward_once
вперёд(n)	robot_forward
назад	robot_back_once
назад(n)	robot_back
поверни налево	robot_left
поверни направо	robot_right
посади	robot_plant
повторить N раз	controls_repeat_ext
повторять, пока	controls_whileUntil
если, то выполнить	controls_if
процедура	procedures_defnoreturn
вызов процедуры	procedures_callnoreturn

7. Редактирование справки

Вы можете для каждого уровня определить сообщение пользователю (инструкцию), которое появляется при загрузке уровня. Найдите в файле `robot_ai.js` массив `HelpContent`:

```
var HelpContent = [ '', // Level 0 not used
// Level 1
  '',
// Level 2
  '',
...
];
```

Если сообщение для какого-то уровня пустое, оно не выводится. Сообщение представляет собой символьную строку, в которой можно использовать HTML-тэги (например, выделять слова жирным и курсивом, вставлять рисунки). Вот пример, в котором на уровне 1 справка содержит таблицу и рисунок:

```
var HelpContent = [ '', // Level 0 not used
// Level 1
  '<table><tr><td><img src="./media/if-block.gif"' +
  'style="margin:10px 15px 0 0;"></td><td>' +
  'Новый блок &laquo;если&raquo; позволяет выполнить группу ' +
  'команд только в том случае, когда верно (истинно) ' +
  'условие после слова &laquo;если&raquo;. ' +
  'Эти команды нужно поставить внутрь блока ' +
  ' в правильном порядке.<br> </td></tr></table>',
// Level 2
  '',
...
];
```

8. Редактирование палитры блоков

Вы можете собрать свою собственную палитру блоков, удалив ненужные блоки. Найдите в конце файла `robot_ai.js` функцию `BlocklyBlocks`:

```
function BlocklyBlocks( Level ) {
  if( [1,2,3,4,5,6,7,8,9,10].includes(Level) ) return '' +
  '<xml xmlns="https://developers.google.com/blockly/xml"' +
  'id="toolbox" style="display: none">' +
//================== Команды Робота ==================
  '<category name="Робот" colour="%{BKY_ROBOT_HUE}">' +
```

```
    // вперед(n)
  '  <block type="robot_forward">' +
  '    <value name="TIMES">' +
  '      <shadow type="math_number">' +
  '        <field name="NUM">1</field>' +
  '      </shadow>' +
  '  </block>' +
    // назад(n)
  '  <block type="robot_back">' +
  '    <value name="TIMES">' +
  '      <shadow type="math_number">' +
  '        <field name="NUM">1</field>' +
  '      </shadow>' +
  '  </block>' +
...
```

Комментарии, которые начинаются с символов `//`, показывают, какой блок далее добавляется. Строчки, где добавляются ненужные вам блоки, можно просто удалить.

В примере, приведённом выше, на всех уровнях используется один и тот же набор блоков. Это не обязательно так. Можно для каждого уровня определить свою палитру блоков:

```
if( [1,2,3,4,5].includes(Level) )
  return '' + ...;
if( [6,7,8,9,10].includes(Level) )
  return '' + ...;
```

В этом примере для уровней 1-5 задаётся один набор блоков, а для уровней 6-10 – другой.

9. Отладка набора задач

Рекомендуется следующая последовательность отладки нового набора задач:

1) подготовить файл `robot_ai.html` и открыть его в браузере Chrome или Firefox, где есть инструменты разработчика;
2) открыть инструменты разработчика (Ctrl+Shift+I), вкладку *Консоль* (*Console*);
3) определить нужный набор блоков;
4) построить карту первого уровня, задать ограничения для него и попробовать обновить страницу;
5) в случае неудачной загрузки посмотреть ошибки на странице *Консоль* инструментов разработчика (обращайте внимание только на ошибки в файле `robot_ai.js`);
6) исправить ошибки и повторить загрузку;
7) после успешной отладки первого уровня перейти ко второму и т.д.