Как создать свой набор задач для Черепахи-Blockly?

Что это такое?

Черепаха-Blockly [1] – это приложение для обучения программированию, в котором исполнитель Черепаха 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 Черепахи, 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

`turtle_ai.html`

`turtle_ai.js`

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

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

`turtle_.html`

`js/turtle_.js`

Создайте их копии с именами `turtle_ai.html` и `turtle_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 `turtle_ai.html`. Найдите в начале файла строки

```
<!-- Измените название вашего приложения  -->
<title>Черепаха: 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/turtle_.js"></script>
<!-- -->
```

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

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

4. Редактирование набора задач для Черепахи

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

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

```
var Maps =
[ {}, // Level 0
  {'x': 0, 'y': 0, 'show': true, 'pen': true, 'width': 2}, // Level 1
  {'x': 0, 'y': 0, 'show': true, 'pen': true, 'width': 2}, // Level 2
  {'x': 0, 'y': 0, 'show': true, 'pen': true, 'width': 2}, // Level 3
  {'x': 0, 'y': 0, 'show': true, 'pen': true, 'width': 2}, // Level 4
  {'x': 0, 'y': 0, 'show': true, 'pen': true, 'width': 2}, // Level 5
  {'x': 0, 'y': 0, 'show': true, 'pen': true, 'width': 2}, // Level 6
  {'x': 0, 'y': 0, 'show': true, 'pen': true, 'width': 2}, // Level 7
  {'x': 0, 'y': 0, 'show': true, 'pen': true, 'width': 2}, // Level 8
  {'x': 0, 'y': 0, 'show': true, 'pen': true, 'width': 2}, // Level 9
  {'x': 0, 'y': 0, 'show': true, 'pen': true, 'width': 2}, // Level 10
];
```

Это массив, в котором каждый уровень описывается как объект. Уровень 0 не используется. Для каждого уровня 1-10 можно определить следующие поля JavaScript-объекта, которые определяют начальное состояние Черепахи:

`x`, `y`: начальные координаты (точка (0,0) – это центр поля Черепахи);

`show`: логическое значение, если оно равно `true`, Черепаха видна на экране;

`pen`: логическое значение, если оно равно `true`, перо Черепахи опущено;

`width`: толщина линии.

Задачи для Черепахи хранятся в виде программ, после выполнения которых на экране появляется нужный рисунок. Эти программы хранятся в массиве `Answer`:

```
var Answers = [ null, // Level 0 not used
// Level 1
function() {
  },
// Level 2
function() {
  },
...
];
```

Самый простой способ построения такой функции-ответа состоит в том, чтобы построить из блоков и отладить нужную программу, затем, используя выпадающий список в нижней части окна, выбрать язык JavaScript, и скопировать всю полученную программу в тело функции-ответа для нужного уровня:

Программа на языке JavaScript

```
1  var a;
2
3  // Определение процедуры
4  function Квадрат(a) {
5    for (var count = 0; count < 4; count++) {
6      forward(a);
7      right(90);
8    }
9    forward(a);
10 }
```

5. Редактирование ограничений

Для каждого уровня вы можете задать максимальное количество блоков, которые можно использовать в решении. Найдите в файле `turtle_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 `вперёд` только два раза. Найдите в файле `turtle_ai.js` массив `someBlockLimit` и добавьте ограничения к уровню `Level 5`:

```
var someBlocksLimit = [ {}, // Level 0 unused
 {  }, // Level 1
 {  }, // Level 2
 {  }, // Level 3
 {  }, // Level 4
 { 'turtle_forward': 2 }, // 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:

`покажись`	`turtle_show`	
`скройся`	`turtle_hide`	
`подними перо`	`turtle_pen_up`	
`опусти перо`	`turtle_pen_down`	
`вперёд`	`turtle_forward_simple`	только число

	turtle_forward	можно с переменной
назад	turtle_back_simple	только число
	turtle_back	можно с переменной
поверни влево	turtle_left	
поверни вправо	turtle_right	
повторить N раз	controls_repeat_list	
процедура	procedures_defnoreturn	
вызов процедуры	procedures_callnoreturn	

6. Редактирование справки

Вы можете для каждого уровня определить сообщение пользователю (инструкцию), которое появляется при загрузке уровня. Найдите в файле `turtle_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
  '',
...
];
```

7. Редактирование палитры блоков

Вы можете собрать свою собственную палитру блоков, удалив ненужные блоки. Найдите в конце файла `turtle_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 Черепахи ==================
  '<category name="Черепаха" colour="%{BKY_ROBOT_HUE}">' +
    // вперед(n)
  '  <block type="turtle_forward">' +
  '    <value name="TIMES">' +
  '      <shadow type="math_number">' +
  '        <field name="NUM">40</field>' +
  '      </shadow>' +
  '    </value>' +
  '  </block>' +
    // назад(n)
  '  <block type="turtle_back">' +
  '    <value name="TIMES">' +
  '      <shadow type="math_number">' +
  '        <field name="NUM">40</field>' +
  '      </shadow>' +
  '    </value>' +
  '  </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.

8. Отладка набора задач

Рекомендуется следующая последовательность отладки нового набора задач:

1) подготовить файл `turtle_ai.html` и открыть его в браузере Chrome или Firefox, где есть инструменты разработчика;
2) определить нужный набор блоков;
3) используя блоки, собрать программу для построения желаемого рисунка для уровня 1;

4) скорректировать начальное положение Черепахи в массиве `Maps` так, чтобы рисунок располагался примерно в центре поля Черепахи;
5) перевести программу на язык JavaScript и добавить решение тело функции нужного уровня в массиве `Answers`;
6) обновить страницу; проверить, чтобы желаемый рисунок был показан на заднем плане;
7) после успешной отладки первого уровня перейти ко второму и т.д.