```
/**
 * Blockly Games: Bird
 *
 * Copyright 2013 Google Inc.
 * https://github.com/google/blockly-games
 *
 * Licensed under the Apache License, Version 2.0 (the "License");
 * you may not use this file except in compliance with the License.
 * You may obtain a copy of the License at
 *
 *   http://www.apache.org/licenses/LICENSE-2.0
 *
 * Unless required by applicable law or agreed to in writing, software
 * distributed under the License is distributed on an "AS IS" BASIS,
 * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
 * See the License for the specific language governing permissions and
 * limitations under the License.
 */

/**
 * @fileoverview JavaScript for Blockly's Bird application.
 * @author fraser@google.com (Neil Fraser)
 */
'use strict';

goog.provide('Bird');

goog.require('Bird.Blocks');
goog.require('Bird.soy');
goog.require('BlocklyDialogs');
goog.require('BlocklyGames');
goog.require('BlocklyInterface');
goog.require('goog.math');
goog.require('goog.math.Coordinate');
goog.require('goog.math.Line');
goog.require('goog.style');


BlocklyGames.NAME = 'bird';

/**
 * Milliseconds between each animation frame.
 */
Bird.stepSpeed;

Bird.BIRD_ICON_SIZE = 120;
Bird.NEST_ICON_SIZE = 100;
Bird.WORM_ICON_SIZE = 100;
Bird.MAP_SIZE = 400;
Bird.WALL_THICKNESS = 10;
Bird.FLAP_SPEED = 100; // ms.

Bird.MAP = [
  // Level 0.
  undefined,
  // Level 1.
  {
    start: new goog.math.Coordinate(20, 20),
    startAngle: 90,
    worm: new goog.math.Coordinate(50, 50),
    nest: new goog.math.Coordinate(80, 80),
    walls: []
  },
  // Level 2.
  {
    start: new goog.math.Coordinate(20, 20),
    startAngle: 0,
    worm: new goog.math.Coordinate(80, 20),
    nest: new goog.math.Coordinate(80, 80),
```

```
70        walls: [new goog.math.Line(0, 50, 60, 50)]
71      },
72      // Level 3.
73      {
74        start: new goog.math.Coordinate(20, 70),
75        startAngle: 270,
76        worm: new goog.math.Coordinate(50, 20),
77        nest: new goog.math.Coordinate(80, 70),
78        walls: [new goog.math.Line(50, 50, 50, 100)]
79      },
80      // Level 4.
81      {
82        start: new goog.math.Coordinate(20, 80),
83        startAngle: 0,
84        worm: null,
85        nest: new goog.math.Coordinate(80, 20),
86        walls: [new goog.math.Line(0, 0, 65, 65)]
87      },
88      // Level 5.
89      {
90        start: new goog.math.Coordinate(80, 80),
91        startAngle: 270,
92        worm: null,
93        nest: new goog.math.Coordinate(20, 20),
94        walls: [new goog.math.Line(0, 100, 65, 35)]
95      },
96      // Level 6.
97      {
98        start: new goog.math.Coordinate(20, 40),
99        startAngle: 0,
100       worm: new goog.math.Coordinate(80, 20),
101       nest: new goog.math.Coordinate(20, 80),
102       walls: [new goog.math.Line(0, 59, 50, 59)]
103     },
104     // Level 7.
105     {
106       start: new goog.math.Coordinate(80, 80),
107       startAngle: 180,
108       worm: new goog.math.Coordinate(80, 20),
109       nest: new goog.math.Coordinate(20, 20),
110       walls: [
111         new goog.math.Line(0, 70, 40, 70),
112         new goog.math.Line(70, 50, 100, 50)
113       ]
114     },
115     // Level 8.
116     {
117       start: new goog.math.Coordinate(20, 25),
118       startAngle: 90,
119       worm: new goog.math.Coordinate(80, 25),
120       nest: new goog.math.Coordinate(80, 75),
121       walls: [
122         new goog.math.Line(50, 0, 50, 25),
123         new goog.math.Line(75, 50, 100, 50),
124         new goog.math.Line(50, 100, 50, 75),
125         new goog.math.Line(0, 50, 25, 50)
126       ]
127     },
128     // Level 9.
129     {
130       start: new goog.math.Coordinate(80, 70),
131       startAngle: 180,
132       worm: new goog.math.Coordinate(20, 20),
133       nest: new goog.math.Coordinate(80, 20),
134       walls: [
135         new goog.math.Line(0, 69, 31, 100),
136         new goog.math.Line(40, 50, 71, 0),
137         new goog.math.Line(80, 50, 100, 50)
138       ]
```

```
    },
    // Level 10.
    {
      start: new goog.math.Coordinate(20, 20),
      startAngle: 90,
      worm: new goog.math.Coordinate(80, 50),
      nest: new goog.math.Coordinate(20, 20),
      walls: [
        new goog.math.Line(40, 60, 60, 60),
        new goog.math.Line(40, 60, 60, 30),
        new goog.math.Line(60, 30, 100, 30)
      ]
    }
  ][BlocklyGames.LEVEL];

  /**
   * PIDs of animation tasks currently executing.
   */
  Bird.pidList = [];

  /**
   * Behaviour for the bird.
   * @enum {number}
   */
  Bird.Pose = {
    SOAR: 1,
    FLAP: 2,
    SIT: 3
  };

  /**
   * Current behaviour.
   * @type Bird.Pose
   */
  Bird.currentPose = Bird.Pose.SOAR;

  /**
   * Create and layout all the nodes for the walls, nest, worm, and bird.
   */
  Bird.drawMap = function() {
    var svg = document.getElementById('svgBird');

    // Add four surrounding walls.
    var edge0 = -Bird.WALL_THICKNESS / 2;
    var edge1 = 100 + Bird.WALL_THICKNESS / 2;
    Bird.MAP.walls.push(new goog.math.Line(edge0, edge0, edge0, edge1));
    Bird.MAP.walls.push(new goog.math.Line(edge0, edge1, edge1, edge1));
    Bird.MAP.walls.push(new goog.math.Line(edge1, edge1, edge1, edge0));
    Bird.MAP.walls.push(new goog.math.Line(edge1, edge0, edge0, edge0));

    // Draw the walls.
    for (var k = 0; k < Bird.MAP.walls.length; k++) {
      var wall = Bird.MAP.walls[k];
      var line = document.createElementNS(Blockly.SVG_NS, 'line');
      line.setAttribute('x1', wall.x0 / 100 * Bird.MAP_SIZE);
      line.setAttribute('y1', (1 - wall.y0 / 100) * Bird.MAP_SIZE);
      line.setAttribute('x2', wall.x1 / 100 * Bird.MAP_SIZE);
      line.setAttribute('y2', (1 - wall.y1 / 100) * Bird.MAP_SIZE);
      line.setAttribute('stroke', '#CCB');
      line.setAttribute('stroke-width', Bird.WALL_THICKNESS);
      line.setAttribute('stroke-linecap', 'round');
      svg.appendChild(line);
    }

    // Add nest.
    var nestImage = document.createElementNS(Blockly.SVG_NS, 'image');
    nestImage.setAttribute('id', 'nest');
    nestImage.setAttributeNS('http://www.w3.org/1999/xlink', 'xlink:href',
        'bird/nest.png');
```

```
    nestImage.setAttribute('height', Bird.NEST_ICON_SIZE);
    nestImage.setAttribute('width', Bird.NEST_ICON_SIZE);
    svg.appendChild(nestImage);

    // Add worm.
    if (Bird.MAP.worm) {
      var birdImage = document.createElementNS(Blockly.SVG_NS, 'image');
      birdImage.setAttribute('id', 'worm');
      birdImage.setAttributeNS('http://www.w3.org/1999/xlink', 'xlink:href',
          'bird/worm.png');
      birdImage.setAttribute('height', Bird.WORM_ICON_SIZE);
      birdImage.setAttribute('width', Bird.WORM_ICON_SIZE);
      svg.appendChild(birdImage);
    }

    // Bird's clipPath element, whose (x, y) is reset by Bird.displayBird
    var birdClip = document.createElementNS(Blockly.SVG_NS, 'clipPath');
    birdClip.setAttribute('id', 'birdClipPath');
    var clipRect = document.createElementNS(Blockly.SVG_NS, 'rect');
    clipRect.setAttribute('id', 'clipRect');
    clipRect.setAttribute('width', Bird.BIRD_ICON_SIZE);
    clipRect.setAttribute('height', Bird.BIRD_ICON_SIZE);
    birdClip.appendChild(clipRect);
    svg.appendChild(birdClip);

    // Add bird.
    var birdIcon = document.createElementNS(Blockly.SVG_NS, 'image');
    birdIcon.setAttribute('id', 'bird');
    birdIcon.setAttributeNS('http://www.w3.org/1999/xlink', 'xlink:href',
        'bird/birds-120.png');
    birdIcon.setAttribute('height', Bird.BIRD_ICON_SIZE * 4); // 120 * 4 = 480
    birdIcon.setAttribute('width', Bird.BIRD_ICON_SIZE * 12); // 120 * 12 = 1440
    birdIcon.setAttribute('clip-path', 'url(#birdClipPath)');
    svg.appendChild(birdIcon);

    // Draw the outer square.
    var square = document.createElementNS(Blockly.SVG_NS, 'rect');
    square.setAttribute('class', 'edges');
    square.setAttribute('width', Bird.MAP_SIZE);
    square.setAttribute('height', Bird.MAP_SIZE);
    svg.appendChild(square);

    var xAxis = BlocklyGames.LEVEL > 3;
    var yAxis = BlocklyGames.LEVEL > 4;

    var TICK_LENGTH = 9;
    var major = 1;
    for (var i = 0.1; i < 0.9; i += 0.1) {
      if (xAxis) {
        // Bottom edge.
        var tick = document.createElementNS(Blockly.SVG_NS, 'line');
        tick.setAttribute('class', 'edges');
        tick.setAttribute('x1', i * Bird.MAP_SIZE);
        tick.setAttribute('y1', Bird.MAP_SIZE);
        tick.setAttribute('x2', i * Bird.MAP_SIZE);
        tick.setAttribute('y2', Bird.MAP_SIZE - TICK_LENGTH * major);
        svg.appendChild(tick);
      }
      if (yAxis) {
        // Left edge.
        var tick = document.createElementNS(Blockly.SVG_NS, 'line');
        tick.setAttribute('class', 'edges');
        tick.setAttribute('x1', 0);
        tick.setAttribute('y1', i * Bird.MAP_SIZE);
        tick.setAttribute('x2', TICK_LENGTH * major);
        tick.setAttribute('y2', i * Bird.MAP_SIZE);
        svg.appendChild(tick);
      }
      if (major == 2) {
```

```
277         if (xAxis) {
278           // X axis.
279           var number = document.createElementNS(Blockly.SVG_NS, 'text');
280           number.setAttribute('class', 'edgeX');
281           number.setAttribute('x', i * Bird.MAP_SIZE + 2);
282           number.setAttribute('y', Bird.MAP_SIZE - 4);
283           number.appendChild(document.createTextNode(Math.round(i * 100)));
284           svg.appendChild(number);
285         }
286         if (yAxis) {
287           // Y axis.
288           var number = document.createElementNS(Blockly.SVG_NS, 'text');
289           number.setAttribute('class', 'edgeY');
290           number.setAttribute('x', 3);
291           number.setAttribute('y', i * Bird.MAP_SIZE - 2);
292           number.appendChild(document.createTextNode(Math.round(100 - i * 100)));
293           svg.appendChild(number);
294         }
295       }
296       major = major == 1 ? 2 : 1;
297     }
298   };
299
300   /**
301    * Initialize Blockly and the bird.  Called on page load.
302    */
303   Bird.init = function() {
304     // Render the Soy template.
305     document.body.innerHTML = Bird.soy.start({}, null,
306         {lang: BlocklyGames.LANG,
307          level: BlocklyGames.LEVEL,
308          maxLevel: BlocklyGames.MAX_LEVEL,
309          html: BlocklyGames.IS_HTML});
310
311     BlocklyInterface.init();
312
313     var rtl = BlocklyGames.isRtl();
314     var blocklyDiv = document.getElementById('blockly');
315     var visualization = document.getElementById('visualization');
316     var onresize = function(e) {
317       var top = visualization.offsetTop;
318       blocklyDiv.style.top = Math.max(10, top - window.pageYOffset) + 'px';
319       blocklyDiv.style.left = rtl ? '10px' : '420px';
320       blocklyDiv.style.width = (window.innerWidth - 440) + 'px';
321     };
322     window.addEventListener('scroll', function() {
323       onresize(null);
324       Blockly.svgResize(BlocklyGames.workspace);
325     });
326     window.addEventListener('resize', onresize);
327     onresize(null);
328
329     var toolbox = document.getElementById('toolbox');
330     BlocklyGames.workspace = Blockly.inject('blockly',
331         {'media': 'third-party/blockly/media/',
332          'rtl': rtl,
333          'toolbox': toolbox,
334          'trashcan': true});
335     BlocklyGames.workspace.getAudioManager().load(
336         ['bird/quack.ogg', 'bird/quack.mp3'], 'quack');
337     BlocklyGames.workspace.getAudioManager().load(
338         ['bird/whack.mp3', 'bird/whack.ogg'], 'whack');
339     BlocklyGames.workspace.getAudioManager().load(
340         ['bird/worm.mp3', 'bird/worm.ogg'], 'worm');
341     if (BlocklyGames.LEVEL > 1) {
342       BlocklyGames.workspace.addChangeListener(Blockly.Events.disableOrphans);
343     }
344     // Not really needed, there are no user-defined functions or variables.
345     Blockly.JavaScript.addReservedWords('noWorm,heading,getX,getY');
```

```
346
347     Bird.drawMap();
348
349     var defaultXml = '';
350     if (BlocklyGames.LEVEL == 1) {
351       defaultXml =
352         '<xml>' +
353         '  <block type="bird_heading" x="70" y="70"></block>' +
354         '</xml>';
355     } else if (BlocklyGames.LEVEL < 5) {
356       defaultXml =
357         '<xml>' +
358         '  <block type="bird_ifElse" x="70" y="70"></block>' +
359         '</xml>';
360     } else {
361       defaultXml =
362         '<xml>' +
363         '  <block type="controls_if" x="70" y="70"></block>' +
364         '</xml>';
365     }
366     BlocklyInterface.loadBlocks(defaultXml, false);
367
368     Bird.reset(true);
369
370     BlocklyGames.bindClick('runButton', Bird.runButtonClick);
371     BlocklyGames.bindClick('resetButton', Bird.resetButtonClick);
372
373     // Open interactive help.  But wait 5 seconds for the
374     // user to think a bit before they are told what to do.
375     setTimeout(function() {
376       BlocklyGames.workspace.addChangeListener(function() {Bird.levelHelp()});
377       Bird.levelHelp();
378     }, 5000);
379     if (BlocklyGames.LEVEL > 8) {
380       setTimeout(BlocklyDialogs.abortOffer, 5 * 60 * 1000);
381     }
382
383     // Lazy-load the JavaScript interpreter.
384     setTimeout(BlocklyInterface.importInterpreter, 1);
385     // Lazy-load the syntax-highlighting.
386     setTimeout(BlocklyInterface.importPrettify, 1);
387   };
388
389   window.addEventListener('load', Bird.init);
390
391   /**
392    * PID of task to poll the mutator's state in level 5.
393    * @private
394    */
395   Bird.mutatorHelpPid_ = 0;
396
397   /**
398    * When the workspace changes, update the help as needed.
399    */
400   Bird.levelHelp = function() {
401     if (BlocklyGames.workspace.isDragging()) {
402       // Don't change helps during drags.
403       return;
404     } else if (BlocklyGames.loadFromLocalStorage(BlocklyGames.NAME,
405                                                  BlocklyGames.LEVEL)) {
406       // The user has already won.  They are just playing around.
407       return;
408     }
409     var rtl = BlocklyGames.isRtl();
410     var userBlocks = Blockly.Xml.domToText(
411         Blockly.Xml.workspaceToDom(BlocklyGames.workspace));
412     var toolbar = BlocklyGames.workspace.flyout_.workspace_.getTopBlocks(true);
413     var content = document.getElementById('dialogHelp');
414     var origin = null;
```

```
415     var style = null;
416     if (BlocklyGames.LEVEL == 1) {
417       if ((userBlocks.indexOf('>90<') != -1 ||
418           userBlocks.indexOf('bird_heading') == -1) &&
419           !Blockly.WidgetDiv.isVisible()) {
420         style = {'width': '370px', 'top': '140px'};
421         style[rtl ? 'right' : 'left'] = '215px';
422         var blocks = BlocklyGames.workspace.getTopBlocks(true);
423         if (blocks.length) {
424           origin = blocks[0].getSvgRoot();
425         } else {
426           origin = toolbar[0].getSvgRoot();
427         }
428       }
429     } else if (BlocklyGames.LEVEL == 2) {
430       if (userBlocks.indexOf('bird_noWorm') == -1) {
431         style = {'width': '350px', 'top': '170px'};
432         style[rtl ? 'right' : 'left'] = '180px';
433         origin = toolbar[1].getSvgRoot();
434       }
435     } else if (BlocklyGames.LEVEL == 4) {
436       if (userBlocks.indexOf('bird_compare') == -1) {
437         style = {'width': '350px', 'top': '230px'};
438         style[rtl ? 'right' : 'left'] = '180px';
439         origin = toolbar[2].getSvgRoot();
440       }
441     } else if (BlocklyGames.LEVEL == 5) {
442       if (!Bird.mutatorHelpPid_) {
443         // Keep polling the mutator's state.
444         Bird.mutatorHelpPid_ = setInterval(Bird.levelHelp, 100);
445       }
446       if (userBlocks.indexOf('mutation else') == -1) {
447         var blocks = BlocklyGames.workspace.getTopBlocks(false);
448         for (var i = 0, block; (block = blocks[i]); i++) {
449           if (block.type == 'controls_if') {
450             break;
451           }
452         }
453         if (!block.mutator.isVisible()) {
454           var xy = goog.style.getPageOffset(block.getSvgRoot());
455           style = {'width': '340px', 'top': (xy.y + 100) + 'px'};
456           style.left = (xy.x - (rtl ? 350 : 0)) + 'px';
457           origin = block.getSvgRoot();
458         } else {
459           content = document.getElementById('dialogMutatorHelp');
460           // Second help box should be below the 'else' block in the mutator.
461           // Really fragile code.  There is no public API for this.
462           origin = block.mutator.workspace_.flyout_.mats_[1];
463           var xy = goog.style.getPageOffset(origin);
464           style = {'width': '340px', 'top': (xy.y + 60) + 'px'};
465           style.left = (xy.x - (rtl ? 310 : 0)) + 'px';
466         }
467       }
468     } else if (BlocklyGames.LEVEL == 6) {
469       if (userBlocks.indexOf('mutation') == -1) {
470         var blocks = BlocklyGames.workspace.getTopBlocks(false);
471         for (var i = 0, block; (block = blocks[i]); i++) {
472           if (block.type == 'controls_if') {
473             break;
474           }
475         }
476         var xy = goog.style.getPageOffset(block.getSvgRoot());
477         style = {'width': '350px', 'top': (xy.y + 220) + 'px'};
478         style.left = (xy.x - (rtl ? 350 : 0)) + 'px';
479         origin = block.getSvgRoot();
480       }
481     } else if (BlocklyGames.LEVEL == 8) {
482       if (userBlocks.indexOf('bird_and') == -1) {
483         style = {'width': '350px', 'top': '360px'};
```

```
484         style[rtl ? 'right' : 'left'] = '450px';
485         origin = toolbar[4].getSvgRoot();
486       }
487     }
488     if (style) {
489       if (content.parentNode != document.getElementById('dialog')) {
490         BlocklyDialogs.showDialog(content, origin, true, false, style, null);
491       }
492     } else {
493       BlocklyDialogs.hideDialog(false);
494     }
495   };
496
497   /**
498    * Reset the bird to the start position and kill any pending animation tasks.
499    * @param {boolean} first True if an opening animation is to be played.
500    */
501   Bird.reset = function(first) {
502     // Kill all tasks.
503     for (var i = 0; i < Bird.pidList.length; i++) {
504       window.clearTimeout(Bird.pidList[i]);
505     }
506     Bird.pidList = [];
507
508     // Move Bird into position.
509     Bird.pos = Bird.MAP.start.clone();
510     Bird.angle = Bird.MAP.startAngle;
511     Bird.currentAngle = Bird.angle;
512     Bird.hasWorm = !Bird.MAP.worm;
513     Bird.currentPose = Bird.Pose.SOAR;
514
515     Bird.displayBird();
516
517     // Move the worm into position.
518     var image = document.getElementById('worm');
519     if (image) {
520       image.setAttribute('x',
521           Bird.MAP.worm.x / 100 * Bird.MAP_SIZE - Bird.WORM_ICON_SIZE / 2);
522       image.setAttribute('y',
523           (1 - Bird.MAP.worm.y / 100) * Bird.MAP_SIZE - Bird.WORM_ICON_SIZE / 2);
524       image.style.visibility = 'visible';
525     }
526     // Move the nest into position.
527     var image = document.getElementById('nest');
528     image.setAttribute('x',
529         Bird.MAP.nest.x / 100 * Bird.MAP_SIZE - Bird.NEST_ICON_SIZE / 2);
530     image.setAttribute('y',
531         (1 - Bird.MAP.nest.y / 100) * Bird.MAP_SIZE - Bird.NEST_ICON_SIZE / 2);
532   };
533
534   /**
535    * Click the run button.  Start the program.
536    * @param {!Event} e Mouse or touch event.
537    */
538   Bird.runButtonClick = function(e) {
539     // Prevent double-clicks or double-taps.
540     if (BlocklyInterface.eventSpam(e)) {
541       return;
542     }
543     var runButton = document.getElementById('runButton');
544     var resetButton = document.getElementById('resetButton');
545     // Ensure that Reset button is at least as wide as Run button.
546     if (!resetButton.style.minWidth) {
547       resetButton.style.minWidth = runButton.offsetWidth + 'px';
548     }
549     runButton.style.display = 'none';
550     resetButton.style.display = 'inline';
551     Bird.reset(false);
552     Bird.execute();
```

```
};

/**
 * Click the reset button.  Reset the bird.
 * @param {!Event} e Mouse or touch event.
 */
Bird.resetButtonClick = function(e) {
  // Prevent double-clicks or double-taps.
  if (BlocklyInterface.eventSpam(e)) {
    return;
  }
  var runButton = document.getElementById('runButton');
  runButton.style.display = 'inline';
  document.getElementById('resetButton').style.display = 'none';
  BlocklyGames.workspace.highlightBlock(null);
  Bird.reset(false);
};

/**
 * Outcomes of running the user program.
 */
Bird.ResultType = {
  UNSET: 0,
  SUCCESS: 1,
  FAILURE: -1,
  TIMEOUT: 2,
  ERROR: -2
};

/**
 * Inject the Bird API into a JavaScript interpreter.
 * @param {!Interpreter} interpreter The JS Interpreter.
 * @param {!Interpreter.Object} scope Global scope.
 */
Bird.initInterpreter = function(interpreter, scope) {
  // API
  var wrapper;
  wrapper = function(angle, id) {
    Bird.heading(angle, id);
  };
  interpreter.setProperty(scope, 'heading',
      interpreter.createNativeFunction(wrapper));
  wrapper = function() {
    return !Bird.hasWorm;
  };
  interpreter.setProperty(scope, 'noWorm',
      interpreter.createNativeFunction(wrapper));
  wrapper = function() {
    return Bird.pos.x;
  };
  interpreter.setProperty(scope, 'getX',
      interpreter.createNativeFunction(wrapper));
  wrapper = function() {
    return Bird.pos.y;
  };
  interpreter.setProperty(scope, 'getY',
      interpreter.createNativeFunction(wrapper));
};

/**
 * Execute the user's code.  Heaven help us...
 */
Bird.execute = function() {
  if (!('Interpreter' in window)) {
    // Interpreter lazy loads and hasn't arrived yet.  Try again later.
    setTimeout(Bird.execute, 250);
    return;
  }

```

```
622     Bird.log = [];
623     Blockly.selected && Blockly.selected.unselect();
624     var code = Blockly.JavaScript.workspaceToCode(BlocklyGames.workspace);
625     var start = code.indexOf('if (');
626     var end = code.indexOf('}\n');
627     if (start != -1 && end != -1) {
628       // Ugly hack: if there is an 'if' statement, ignore isolated heading blocks.
629       code = code.substring(start, end + 2);
630     }
631     code = 'while(true) {\n' +
632         code +
633         '}';
634     var result = Bird.ResultType.UNSET;
635     var interpreter = new Interpreter(code, Bird.initInterpreter);
636
637     // Try running the user's code.  There are four possible outcomes:
638     // 1. If bird reaches the finish [SUCCESS], true is thrown.
639     // 2. If the program is terminated due to running too long [TIMEOUT],
640     //    false is thrown.
641     // 3. If another error occurs [ERROR], that error is thrown.
642     // 4. If the program ended normally but without finishing [FAILURE],
643     //    no error or exception is thrown.
644     try {
645       var ticks = 100000;  // 100k ticks runs Bird for about 3 minutes.
646       while (interpreter.step()) {
647         if (ticks-- <= 0) {
648           throw Infinity;
649         }
650       }
651       result = Bird.ResultType.FAILURE;
652     } catch (e) {
653       // A boolean is thrown for normal termination.
654       // Abnormal termination is a user error.
655       if (e === Infinity) {
656         result = Bird.ResultType.TIMEOUT;
657       } else if (e === true) {
658         result = Bird.ResultType.SUCCESS;
659       } else if (e === false) {
660         result = Bird.ResultType.ERROR;
661       } else {
662         // Syntax error, can't happen.
663         result = Bird.ResultType.ERROR;
664         window.alert(e);
665       }
666     }
667
668     // Fast animation if execution is successful.  Slow otherwise.
669     Bird.stepSpeed = (result == Bird.ResultType.SUCCESS) ? 10 : 15;
670
671     // Bird.log now contains a transcript of all the user's actions.
672     // Reset the bird and animate the transcript.
673     Bird.reset(false);
674     Bird.pidList.push(setTimeout(Bird.animate, 1));
675   };
676
677   /**
678    * Iterate through the recorded path and animate the bird's actions.
679    */
680   Bird.animate = function() {
681     // All tasks should be complete now.  Clean up the PID list.
682     Bird.pidList = [];
683
684     var action = Bird.log.shift();
685     if (!action) {
686       BlocklyInterface.highlight(null);
687       return;
688     }
689     BlocklyInterface.highlight(action.pop());
690
```

```
    if (action[0] == 'move' || action[0] == 'goto') {
      Bird.pos.x = action[1];
      Bird.pos.y = action[2];
      Bird.angle = action[3];
      Bird.currentPose = action[0] == 'move' ? Bird.Pose.FLAP : Bird.Pose.SOAR;
      Bird.displayBird();
    } else if (action[0] == 'worm') {
      var worm = document.getElementById('worm');
      worm.style.visibility = 'hidden';
    } else if (action[0] == 'finish') {
      Bird.currentPose = Bird.Pose.SIT;
      Bird.displayBird();
      BlocklyInterface.saveToLocalStorage();
      BlocklyDialogs.congratulations();
    } else if (action[0] == 'play') {
      BlocklyGames.workspace.getAudioManager().play(action[1], 0.5);
    }

    Bird.pidList.push(setTimeout(Bird.animate, Bird.stepSpeed * 5));
  };

  /**
   * Display bird at the current location, facing the current angle.
   */
  Bird.displayBird = function() {
    var diff = goog.math.angleDifference(Bird.angle, Bird.currentAngle);
    var step = 10;
    if (Math.abs(diff) <= step) {
      Bird.currentAngle = Bird.angle;
    } else {
      Bird.currentAngle -= goog.math.sign(diff) * step;
      Bird.currentAngle = goog.math.standardAngle(Bird.currentAngle);
    }
    // Divide into 12 quads.
    var quad = (14 - Math.round(Bird.currentAngle / 360 * 12)) % 12;
    var quadAngle = 360 / 12;  // 30.
    var remainder = Bird.currentAngle % quadAngle;
    if (remainder >= quadAngle / 2) {
      remainder -= quadAngle;
    }
    remainder *= -1;

    var row;
    if (Bird.currentPose == Bird.Pose.SOAR) {
      row = 0;
    } else if (Bird.currentPose == Bird.Pose.SIT) {
      row = 3;
    } else if (Bird.currentPose == Bird.Pose.FLAP) {
      row = Math.round(Date.now() / Bird.FLAP_SPEED) % 3;
    } else {
      throw 'Unknown pose.';
    }

    var x = Bird.pos.x / 100 * Bird.MAP_SIZE - Bird.BIRD_ICON_SIZE / 2;
    var y = (1 - Bird.pos.y / 100) * Bird.MAP_SIZE - Bird.BIRD_ICON_SIZE / 2;
    var birdIcon = document.getElementById('bird');
    birdIcon.setAttribute('x', x - quad * Bird.BIRD_ICON_SIZE);
    birdIcon.setAttribute('y', y - row * Bird.BIRD_ICON_SIZE);
    birdIcon.setAttribute('transform', 'rotate(' + remainder + ', ' +
        (x + Bird.BIRD_ICON_SIZE / 2) + ', ' +
        (y + Bird.BIRD_ICON_SIZE / 2) + ')');

    var clipRect = document.getElementById('clipRect');
    clipRect.setAttribute('x', x);
    clipRect.setAttribute('y', y);
  };

  /**
   * Has the bird intersected the nest?
```

```
760     * @return {boolean} True if the bird found the nest, false otherwise.
761     */
762   Bird.intersectNest = function() {
763     var accuracy = 0.5 * Bird.BIRD_ICON_SIZE / Bird.MAP_SIZE * 100;
764     return goog.math.Coordinate.distance(Bird.pos, Bird.MAP.nest) < accuracy;
765   };
766
767   /**
768    * Has the bird intersected the worm?
769    * @return {boolean} True if the bird found the worm, false otherwise.
770    */
771   Bird.intersectWorm = function() {
772     if (Bird.MAP.worm) {
773       var accuracy = 0.5 * Bird.BIRD_ICON_SIZE / Bird.MAP_SIZE * 100;
774       return goog.math.Coordinate.distance(Bird.pos, Bird.MAP.worm) < accuracy;
775     }
776     return false;
777   };
778
779   /**
780    * Has the bird intersected a wall?
781    * @return {boolean} True if the bird hit a wall, false otherwise.
782    */
783   Bird.intersectWall = function() {
784     var accuracy = 0.2 * Bird.BIRD_ICON_SIZE / Bird.MAP_SIZE * 100;
785     for (var i = 0, wall; (wall = Bird.MAP.walls[i]); i++) {
786       var wallPoint = wall.getClosestSegmentPoint(Bird.pos);
787       if (goog.math.Coordinate.distance(wallPoint, Bird.pos) < accuracy) {
788         return true;
789       }
790     }
791     return false;
792   };
793
794   /**
795    * Move the bird to the given point.
796    * @param {!goog.math.Coordinate} p Coordinate of point.
797    */
798   Bird.gotoPoint = function(p) {
799     var steps = Math.round(goog.math.Coordinate.distance(Bird.pos, p));
800     var angleDegrees = goog.math.angle(Bird.pos.x, Bird.pos.y, p.x, p.y);
801     var angleRadians = goog.math.toRadians(angleDegrees);
802     for (var i = 0; i < steps; i++) {
803       Bird.pos.x += Math.cos(angleRadians);
804       Bird.pos.y += Math.sin(angleRadians);
805       Bird.log.push(['goto', Bird.pos.x, Bird.pos.y, angleDegrees, null]);
806     }
807   };
808
809   /**
810    * Attempt to move the bird in the specified direction.
811    * @param {number} angle Direction to move (0 = east, 90 = north).
812    * @param {string} id ID of block that triggered this action.
813    * @throws {true} If the nest is reached.
814    * @throws {false} If the bird collides with a wall.
815    */
816   Bird.heading = function(angle, id) {
817     var angleRadians = goog.math.toRadians(angle);
818     Bird.pos.x += Math.cos(angleRadians);
819     Bird.pos.y += Math.sin(angleRadians);
820     Bird.angle = angle;
821     Bird.log.push(['move', Bird.pos.x, Bird.pos.y, Bird.angle, id]);
822     if (Bird.hasWorm && Bird.intersectNest()) {
823       // Finished.  Terminate the user's program.
824       Bird.log.push(['play', 'quack', null]);
825       Bird.gotoPoint(Bird.MAP.nest);
826       Bird.log.push(['finish', null]);
827       throw true;
828     }
```

```
    if (!Bird.hasWorm && Bird.intersectWorm()) {
      Bird.gotoPoint(Bird.MAP.worm);
      Bird.log.push(['worm', null]);
      Bird.log.push(['play', 'worm', null]);
      Bird.hasWorm = true;
    }
    if (Bird.intersectWall()) {
      Bird.log.push(['play', 'whack', null]);
      throw false;
    }
  };

```