dancevis.Shapes.ShapeTypeld class specification

Constant

<table>
<thead>
<tr>
<th>Constant</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>LINE</td>
<td>The type for a line shape.</td>
</tr>
<tr>
<td>CIRCLE</td>
<td>The type for a circle shape.</td>
</tr>
<tr>
<td>GRID</td>
<td>The type for a grid shape.</td>
</tr>
</tbody>
</table>

Static Methods

<table>
<thead>
<tr>
<th>Static Methods</th>
<th>Return Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>isValidShapeType(typeId: ShapeTypeld)</td>
<td>boolean</td>
<td>Is the shape type valid.</td>
</tr>
</tbody>
</table>

dancevis.Shapes.GeometricShape class

Constructor

<table>
<thead>
<tr>
<th>Constructor</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GeometricShape(shapeTypeld: ShapeTypeld)</td>
<td>Base class for all shapes.</td>
</tr>
</tbody>
</table>

Properties

<table>
<thead>
<tr>
<th>Properties</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>shapeld</td>
<td>number</td>
<td>A unique integer to identify this shape.</td>
</tr>
<tr>
<td>shapeTypeld</td>
<td>ShapeTypeld</td>
<td>The shape type.</td>
</tr>
</tbody>
</table>
dancevis.Shapes.Line class

Extends GeometricShape class.

Constructor

<table>
<thead>
<tr>
<th>Constructor</th>
<th>Description</th>
</tr>
</thead>
</table>

Methods

<table>
<thead>
<tr>
<th>Methods</th>
<th>Return Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>startPosition()</td>
<td>Position</td>
<td>Returns the starting position of the line.</td>
</tr>
<tr>
<td>endPosition()</td>
<td>Position</td>
<td>Returns the ending position of the line.</td>
</tr>
<tr>
<td>getOrientation()</td>
<td>Orientation</td>
<td>Returns the orientation of the line.</td>
</tr>
<tr>
<td>setOrientation(orientation: Orientation)</td>
<td>None</td>
<td>Sets the orientation of the line.</td>
</tr>
<tr>
<td>getPosition()</td>
<td>Position</td>
<td>Returns the start position of the line.</td>
</tr>
<tr>
<td>setPosition(position: Position)</td>
<td>None</td>
<td>Sets the start position of the line.</td>
</tr>
<tr>
<td>nextPositionAndOrientation(startPosition: Position, dt: Time, speed: Speed)</td>
<td>Position, Orientation</td>
<td>Returns a new position and orientation that represents how an object moves along the line.</td>
</tr>
<tr>
<td>length()</td>
<td>number</td>
<td>Return the length of the line.</td>
</tr>
<tr>
<td>angle()</td>
<td>Orientation</td>
<td>Return the angle of the line.</td>
</tr>
<tr>
<td>isOnShape(position: Position)</td>
<td>boolean</td>
<td>Return true if this position is on the line.</td>
</tr>
<tr>
<td>showShapeOnScreen(bool: boolean)</td>
<td>None</td>
<td>Show shape on screen or don’t show it.</td>
</tr>
<tr>
<td>copy()</td>
<td>Line</td>
<td>Return a new line that is a copy.</td>
</tr>
</tbody>
</table>

Properties

<table>
<thead>
<tr>
<th>Properties</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>length</td>
<td>number</td>
<td>Length of the line</td>
</tr>
<tr>
<td>angle</td>
<td>Orientation</td>
<td>Angle of the line.</td>
</tr>
</tbody>
</table>
dancevis.Shapes.Circle class

Extends GeometricShape class.

Constructor

<table>
<thead>
<tr>
<th>Constructor</th>
<th>Description</th>
</tr>
</thead>
</table>

Methods

<table>
<thead>
<tr>
<th>Methods</th>
<th>Return Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>getPosition()</td>
<td>Position</td>
<td>Return the center of this circle.</td>
</tr>
<tr>
<td>setPosition(position:Position)</td>
<td>None</td>
<td>Set the center of this circle.</td>
</tr>
<tr>
<td>getOrientation()</td>
<td>Orientation</td>
<td>Return the orientation of the circle.</td>
</tr>
<tr>
<td>setOrientation(angle:Orientation)</td>
<td>None</td>
<td>Set the orientation of the circle.</td>
</tr>
<tr>
<td>nextPositionAndOrientation(startPosition:Position, dt:Time, speed:Speed)</td>
<td>Position, Orientation</td>
<td>Returns a new position and orientation that represents how an object moves along the circle.</td>
</tr>
<tr>
<td>positionAtAngle(angle:Orientation)</td>
<td>Position</td>
<td>Return the position at that angle on the circle.</td>
</tr>
<tr>
<td>angleFromPosition(position:Position)</td>
<td>Orientation</td>
<td>Return the angle that this position represents.</td>
</tr>
<tr>
<td>arcLength(angle:Orientation)</td>
<td>number</td>
<td>Return the length of an arc that spans the angle.</td>
</tr>
<tr>
<td>setStartAngle(angle:Orientation)</td>
<td>None</td>
<td>Set the start angle.</td>
</tr>
<tr>
<td>setStopAngle(angle:Orientation)</td>
<td>None</td>
<td>Set the angle where we stop.</td>
</tr>
<tr>
<td>startAngle()</td>
<td>number</td>
<td>Return the start angle of the circle.</td>
</tr>
<tr>
<td>stopAngle()</td>
<td>number</td>
<td>Return the stop angle of the circle.</td>
</tr>
<tr>
<td>isOnShape(position:Position)</td>
<td>boolean</td>
<td>Return true if this position is on the circle.</td>
</tr>
<tr>
<td>showShapeOnScreen(bool:boolean)</td>
<td>None</td>
<td>Show shape on screen or don’t show it.</td>
</tr>
<tr>
<td>copy()</td>
<td>Circle</td>
<td>Return a new circle that is a copy.</td>
</tr>
<tr>
<td>Properties</td>
<td>Type</td>
<td>Description</td>
</tr>
<tr>
<td>-------------</td>
<td>-----------</td>
<td>-----------------------------------------</td>
</tr>
<tr>
<td>radius</td>
<td>number</td>
<td>Radius of the circle.</td>
</tr>
<tr>
<td>center</td>
<td>Position</td>
<td>Center position of the circle.</td>
</tr>
<tr>
<td>startAngle</td>
<td>Orientation</td>
<td>Angle at which we start.</td>
</tr>
<tr>
<td>stopAngle</td>
<td>Orientation</td>
<td>Angle at which to stop.</td>
</tr>
<tr>
<td>clockwise</td>
<td>boolean</td>
<td>This is a clockwise circle.</td>
</tr>
</tbody>
</table>
dancevis.Shapes.Grid class

Extends GeometricShape class.

Constructor

<table>
<thead>
<tr>
<th>Constructor</th>
<th>Description</th>
</tr>
</thead>
</table>

Methods

<table>
<thead>
<tr>
<th>Methods</th>
<th>Return Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>positionAt(row:number, col:number)</td>
<td>Position</td>
<td>Returns the position at that (row, col).</td>
</tr>
<tr>
<td>numRows()</td>
<td>number</td>
<td>Return the number of rows in the grid.</td>
</tr>
<tr>
<td>numCols()</td>
<td>number</td>
<td>Return the number of columns in the grid.</td>
</tr>
<tr>
<td>setCenter(center:Position)</td>
<td>None</td>
<td>Set the center position of the grid.</td>
</tr>
<tr>
<td>isOnShape(position:Position)</td>
<td>boolean</td>
<td>Returns true if the given position is at any of the positions specified by this grid.</td>
</tr>
<tr>
<td>showShapeOnScreen(bool:boolean)</td>
<td>None</td>
<td>Show shape on screen or don't show it.</td>
</tr>
</tbody>
</table>

Properties

<table>
<thead>
<tr>
<th>Properties</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>numRows</td>
<td>number</td>
<td>The number of rows.</td>
</tr>
<tr>
<td>numCols</td>
<td>number</td>
<td>The number of columns</td>
</tr>
</tbody>
</table>
dancevis.GroupExitPoint object specification

Properties

<table>
<thead>
<tr>
<th>Properties</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>startTime</td>
<td>Time</td>
<td>The time when this exit point becomes active.</td>
</tr>
<tr>
<td>endTime</td>
<td>Time</td>
<td>The time when this exit point becomes inactive.</td>
</tr>
<tr>
<td>position</td>
<td>Position</td>
<td>The position on the group where the children will exit.</td>
</tr>
<tr>
<td>nextGroup</td>
<td>Group</td>
<td>The group the children will exit to.</td>
</tr>
<tr>
<td>name</td>
<td>String</td>
<td>Optional name for this exit point. If no name is provided, then the string</td>
</tr>
<tr>
<td></td>
<td></td>
<td>value of the position will be used.</td>
</tr>
<tr>
<td>showOnScreen</td>
<td>boolean</td>
<td>True or false if we display this exit point on screen. Default to true.</td>
</tr>
</tbody>
</table>

dancevis.GroupOptions object specification

Properties

<table>
<thead>
<tr>
<th>Properties</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>shape</td>
<td>Shape</td>
<td>The shape associated with this group. Required.</td>
</tr>
<tr>
<td>startTime</td>
<td>Time</td>
<td>The time at which this group becomes active. Required.</td>
</tr>
<tr>
<td>endTime</td>
<td>Time</td>
<td>The time at which this group becomes inactive. Required.</td>
</tr>
<tr>
<td>parentGroup</td>
<td>Group</td>
<td>The parent of this group.</td>
</tr>
<tr>
<td>position</td>
<td>Position</td>
<td>The group position.</td>
</tr>
<tr>
<td>speed</td>
<td>Speed</td>
<td>The speed at which children move in this group.</td>
</tr>
</tbody>
</table>
dancevis.GroupClientUpdateFunction object specification

<table>
<thead>
<tr>
<th>Properties</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>begin</td>
<td>Time</td>
<td>The time when this function will start being applied. Required.</td>
</tr>
<tr>
<td>end</td>
<td>Time</td>
<td>The time when this function will stop being applied. Required.</td>
</tr>
</tbody>
</table>
dancevis.Group class

### Constructor

<table>
<thead>
<tr>
<th>Constructor</th>
<th>Description</th>
</tr>
</thead>
</table>
| Group(groupOptions?:
  |GroupOptions)                    | Creates a new group              |

### Methods

<table>
<thead>
<tr>
<th>Methods</th>
<th>Return Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>getGroupName()</td>
<td>string</td>
<td>The name of this group.</td>
</tr>
<tr>
<td>setGroupName(name:string)</td>
<td>None</td>
<td>Set the name of this group.</td>
</tr>
<tr>
<td>updateChildrenBasedOnMyShape(currentTime:Time)</td>
<td>None</td>
<td>Updates the children of this group according to this group’s shape. Then call updateChildrenBasedOnMyShape on every child of this group passing in the current time.</td>
</tr>
<tr>
<td>setMyPositionAndModifyChildren(position:Position)</td>
<td>None</td>
<td>Modifies the position of this group and then recursively updates position of all children.</td>
</tr>
<tr>
<td>childrenTimels(currenttime:Time)</td>
<td>None</td>
<td>Notify children of the current time.</td>
</tr>
<tr>
<td>timeIs(currentTime:Time)</td>
<td>None</td>
<td>Update the group and its children to be consistent with the current time.</td>
</tr>
<tr>
<td>insertChild(child:Group</td>
<td>Dancer, index?:number)</td>
<td>None</td>
</tr>
<tr>
<td>removeChild(index:number)</td>
<td>Group</td>
<td>Dancer</td>
</tr>
<tr>
<td>removeChildById(groupId:number)</td>
<td>None</td>
<td>Remove a child.</td>
</tr>
<tr>
<td>addExitPoint(groupEPObj:Group ExitPoint)</td>
<td>None</td>
<td>Add and exit point to this group.</td>
</tr>
<tr>
<td>getOrientation()</td>
<td>Orientation</td>
<td>Return the orientation for this group.</td>
</tr>
<tr>
<td>setOrientation(angle:Orientation)</td>
<td>None</td>
<td>Set the orientation for this group.</td>
</tr>
</tbody>
</table>
**setParent**(parent: `Group`)  None  Set the parent of this group to be the given group.
Note: this also adds this group as a child of the given group.

**setBeginAction**(func: `Function.<Group|Dancer>()`)  None  Set the function that is called on each child before this group becomes active.

**setEndAction**(func: `Function.<Group|Dancer>()`)  None  Set the function that is called on each child before this group becomes inactive.

**setEndCondition**(func: `Function.<Group|Dancer>()`)  None  Set the function that checks if a child of this group has reached the end condition.

**showShapeOnScreen**(bool: `boolean`)  None  Show shape on screen or don’t show it.

**getPosition**()  `Position`  Return the position of this group.

**setPosition**(position: `Position`)  None  Set the position for this group

---

**Properties**

<table>
<thead>
<tr>
<th>Properties</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>children</td>
<td>Array.&lt;<code>Group</code></td>
<td><code>Dancer</code>&gt;</td>
</tr>
<tr>
<td>parentGroup</td>
<td><code>Group</code></td>
<td>Parent group</td>
</tr>
<tr>
<td>shape</td>
<td><code>Shape</code></td>
<td>A geometric object</td>
</tr>
<tr>
<td>speed</td>
<td><code>Speed</code></td>
<td>Speed of group</td>
</tr>
<tr>
<td>startTime</td>
<td><code>Time</code></td>
<td>Time at which this group becomes active</td>
</tr>
<tr>
<td>endTime</td>
<td><code>Time</code></td>
<td>Time at which this group becomes inactive</td>
</tr>
<tr>
<td>lastTime</td>
<td><code>Time</code></td>
<td>The last time that this group was updated for.</td>
</tr>
<tr>
<td>active</td>
<td>boolean</td>
<td>True if this group is active.</td>
</tr>
<tr>
<td>position</td>
<td><code>Position</code></td>
<td>The position of this group.</td>
</tr>
<tr>
<td>onChildAddition</td>
<td>`Function.&lt;Group</td>
<td>Dancer&gt;()`</td>
</tr>
<tr>
<td>onChildRemoval</td>
<td>Function.&lt;Group</td>
<td>Dancer&gt;()</td>
</tr>
<tr>
<td>-------------------</td>
<td>-------------------------------------------------</td>
<td></td>
</tr>
<tr>
<td>endCondition</td>
<td>Function.&lt;Group</td>
<td>Dancer&gt;()</td>
</tr>
<tr>
<td>clientUpdateFunctions</td>
<td>Array.&lt; Function.&lt;Group</td>
<td>Dancer&gt; &gt;</td>
</tr>
</tbody>
</table>
# dancevis.TimeSet object specification

## Properties

<table>
<thead>
<tr>
<th>Properties</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>milliseconds</td>
<td>number</td>
<td>The count of milliseconds going into this time.</td>
</tr>
<tr>
<td>seconds</td>
<td>number</td>
<td>The count of seconds going into this time.</td>
</tr>
<tr>
<td>minutes</td>
<td>number</td>
<td>The count of minutes going into this time.</td>
</tr>
<tr>
<td>hours</td>
<td>number</td>
<td>The count of hours going into this time.</td>
</tr>
</tbody>
</table>

## dancevis.Time class

### Constructor

<table>
<thead>
<tr>
<th>Constructor</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Time(timeSet?: TimeSet)</td>
<td>Creates a new Time object. Defaults to time zero.</td>
</tr>
</tbody>
</table>

### Methods

<table>
<thead>
<tr>
<th>Methods</th>
<th>Return Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>inMilliseconds()</td>
<td>number</td>
<td>This time represented as a count of milliseconds.</td>
</tr>
<tr>
<td>inSeconds()</td>
<td>number</td>
<td>This time represented as a count of seconds.</td>
</tr>
<tr>
<td>inMinutes()</td>
<td>number</td>
<td>This time represented as a count of minutes.</td>
</tr>
<tr>
<td>inHours()</td>
<td>number</td>
<td>This time represented as a count of hours.</td>
</tr>
<tr>
<td>inTimeUnits()</td>
<td>Object</td>
<td>Return the current time split into minutes, seconds, milliseconds.</td>
</tr>
<tr>
<td>equals(other: Time)</td>
<td>boolean</td>
<td>Compares two times.</td>
</tr>
<tr>
<td>isBetween(t1: Time, t2: Time)</td>
<td>boolean</td>
<td>Return true if this time is between the two given times.</td>
</tr>
<tr>
<td>toString()</td>
<td>string</td>
<td>The string representation of this time.</td>
</tr>
<tr>
<td>copy()</td>
<td>Time</td>
<td>Return a copy of this time.</td>
</tr>
</tbody>
</table>
## Properties

<table>
<thead>
<tr>
<th>Properties</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>milliseconds</td>
<td>number</td>
<td>The number of milliseconds in this time.</td>
</tr>
</tbody>
</table>

## Static Variables

<table>
<thead>
<tr>
<th>Static Variables</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>zeroTime</td>
<td>Date</td>
<td>The date object that represents when time zero is.</td>
</tr>
</tbody>
</table>

## Static Methods

<table>
<thead>
<tr>
<th>Static Methods</th>
<th>Return Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>zeroTimeIsNow()</td>
<td>None</td>
<td>Set the zero time for the simulation.</td>
</tr>
<tr>
<td>now()</td>
<td>Time</td>
<td>Return the current time as an offset from the time when zero time was set.</td>
</tr>
</tbody>
</table>
dancevis.SpeedSet object specification

Properties

<table>
<thead>
<tr>
<th>Properties</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>startPosition</td>
<td>Position</td>
<td>The starting position.</td>
</tr>
<tr>
<td>endPosition</td>
<td>Position</td>
<td>The ending position.</td>
</tr>
<tr>
<td>duration</td>
<td>Time</td>
<td>The length of time needed to travel from start position to end position. This cannot be 0.</td>
</tr>
<tr>
<td>distance</td>
<td>number</td>
<td>The distance to be traversed in duration time. Note that either distance OR startPosition/endPosition should be specified, but distance takes priority.</td>
</tr>
<tr>
<td>speed</td>
<td>number</td>
<td>The speed to use. This takes priority over all other settings.</td>
</tr>
</tbody>
</table>

dancevis.Speed class

Constructor

<table>
<thead>
<tr>
<th>Constructor</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Speed(speedSet?: SpeedSet)</td>
<td>Creates a new Speed object. Defaults to speed 1.</td>
</tr>
</tbody>
</table>

Methods

<table>
<thead>
<tr>
<th>Methods</th>
<th>Return Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>speed()</td>
<td>number</td>
<td>Return the speed</td>
</tr>
<tr>
<td>setSpeed(speedSet: SpeedSet)</td>
<td>None</td>
<td>Set the speed.</td>
</tr>
<tr>
<td>equals(other: Speed)</td>
<td>boolean</td>
<td>Compares two speeds.</td>
</tr>
<tr>
<td>toString()</td>
<td>string</td>
<td>The string representation of this speed.</td>
</tr>
<tr>
<td>copy()</td>
<td>Speed</td>
<td>Return a copy of this speed.</td>
</tr>
</tbody>
</table>

Properties
pixelsPerSecond number The speed.

dancevis.Position class

Constructor

<table>
<thead>
<tr>
<th>Constructor</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Position(x:number, y:number)</td>
<td>Creates a Position object representing a point on the screen where the origin is in the center of the screen and y increases as you go towards the top of the screen.</td>
</tr>
</tbody>
</table>

Methods

<table>
<thead>
<tr>
<th>Methods</th>
<th>Return Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>distance(other:Position)</td>
<td>number</td>
<td>The euclidian distance between this position and the given position.</td>
</tr>
<tr>
<td>positionInDirection(distance:number, angle:Orientation)</td>
<td>Position</td>
<td>Return the position that is distance away from this position in the direction of the given angle.</td>
</tr>
<tr>
<td>equals(other:Position)</td>
<td>boolean</td>
<td>Compares two positions.</td>
</tr>
<tr>
<td>copy()</td>
<td>Position</td>
<td>Returns a copy of this position.</td>
</tr>
<tr>
<td>screenCoords()</td>
<td>Position</td>
<td>Returns a new position with x and y converted to screen coordinates.</td>
</tr>
<tr>
<td>toString()</td>
<td>string</td>
<td>Returns a string representation of this Position.</td>
</tr>
</tbody>
</table>

Properties

<table>
<thead>
<tr>
<th>Properties</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>x</td>
<td>number</td>
<td>The x position.</td>
</tr>
<tr>
<td>y</td>
<td>number</td>
<td>The y position.</td>
</tr>
</tbody>
</table>

Static Variables
### Static Methods

<table>
<thead>
<tr>
<th>Static Methods</th>
<th>Return Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>screenOriginIs(left:number, top:number)</td>
<td>None</td>
<td>Set the origin of our coordinate system at this screen position in the browser.</td>
</tr>
<tr>
<td>screenToModelCoords(left:number, top:number)</td>
<td>Position</td>
<td>Return the screen position in our model coordinate system.</td>
</tr>
</tbody>
</table>
### dancevis.Orientation class

#### Constructor

<table>
<thead>
<tr>
<th>Constructor</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Orientation(angle:number, isRadians?:boolean)</td>
<td>Creates an Orientation object representing an angle. The angle defaults to 0 radians.</td>
</tr>
</tbody>
</table>

#### Methods

<table>
<thead>
<tr>
<th>Methods</th>
<th>Return Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>inRadians()</td>
<td>number</td>
<td>Return the the angle in radians.</td>
</tr>
<tr>
<td>inDegrees()</td>
<td>number</td>
<td>Return the angle in degrees.</td>
</tr>
<tr>
<td>equals(other: Orientation)</td>
<td>boolean</td>
<td>Compares two orientations.</td>
</tr>
<tr>
<td>cos()</td>
<td>number</td>
<td>Returns cosine of this orientation.</td>
</tr>
<tr>
<td>sin()</td>
<td>number</td>
<td>Returns sine of this orientation.</td>
</tr>
<tr>
<td>isBetween(o1: Orientation, o2: Orientation)</td>
<td>boolean</td>
<td>Return true if this orientation is between the two given ones.</td>
</tr>
<tr>
<td>angleBetween(other: Orientation)</td>
<td>Orientation</td>
<td>Returns the smallest angle between</td>
</tr>
<tr>
<td>toString()</td>
<td>string</td>
<td>Returns a string representation of this Orientation.</td>
</tr>
<tr>
<td>copy()</td>
<td>Orientation</td>
<td>Return a copy of this orientation.</td>
</tr>
</tbody>
</table>

#### Properties

<table>
<thead>
<tr>
<th>Properties</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>angle</td>
<td>number</td>
<td>The angle.</td>
</tr>
</tbody>
</table>

#### Static Methods
<table>
<thead>
<tr>
<th>Function</th>
<th>Parameters</th>
<th>Returns</th>
</tr>
</thead>
<tbody>
<tr>
<td>radiansToDegrees(radians:number)</td>
<td>number</td>
<td>Returns degree equivalent of angle in radians.</td>
</tr>
<tr>
<td>degreesToRadians(degrees:number)</td>
<td>number</td>
<td>Returns radian equivalent of an angle in degrees.</td>
</tr>
<tr>
<td>positionsToOrientation(pos1:Position, pos2:Position)</td>
<td>Orientation</td>
<td>Return a new orientation from the two given positions.</td>
</tr>
</tbody>
</table>
**dancevis.DancerTypeld class**

Identifiers for common dancer types.

**Constant**

<table>
<thead>
<tr>
<th>Constant</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>FOLLOW</td>
<td>This dancer type represents a follow.</td>
</tr>
<tr>
<td>LEAD</td>
<td>This dancer type represents a lead.</td>
</tr>
</tbody>
</table>

**dancevis.DancerShape class**

Identifiers for common dancer shapes.

**Constant**

<table>
<thead>
<tr>
<th>Constant</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CIRCLE</td>
<td>Use a circle on screen to represent a dancer.</td>
</tr>
</tbody>
</table>

**dancevis.DancerShapeSize class**

Identifiers for common dancer sizes.

**Constant**

<table>
<thead>
<tr>
<th>Constant</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SMALL</td>
<td>Make the dancer shape small.</td>
</tr>
<tr>
<td>MEDIUM</td>
<td>Make the dancer shape medium sized.</td>
</tr>
<tr>
<td>LARGE</td>
<td>Make the dancer shape large.</td>
</tr>
</tbody>
</table>
**dancevis.DancerOptions object specification**

## Properties

<table>
<thead>
<tr>
<th>Properties</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>dancerTypeId</td>
<td>DancerTypeId</td>
<td>The dancer type.</td>
</tr>
<tr>
<td>dancerShape</td>
<td>DancerShape</td>
<td>The shape that is displayed on screen to represent the dancer.</td>
</tr>
<tr>
<td>dancerSize</td>
<td>DancerShapeSize</td>
<td>The size of the dancer on screen.</td>
</tr>
<tr>
<td>dancerName</td>
<td>string</td>
<td>The name of this dancer.</td>
</tr>
<tr>
<td>dancerColor</td>
<td>string</td>
<td>The color for the underlying element that is bound to this dancer.</td>
</tr>
<tr>
<td>position</td>
<td>Position</td>
<td>The initial position of the dancer. Optional.</td>
</tr>
<tr>
<td>orientation</td>
<td>Orientation</td>
<td>The current orientation of the dancer. Optional.</td>
</tr>
</tbody>
</table>
# dancevis.Dancer class

## Constructor

<table>
<thead>
<tr>
<th>Constructor</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dancer(dancerOptions?::DancerOptions)</td>
<td>Creates a new dancer.</td>
</tr>
</tbody>
</table>

## Methods

<table>
<thead>
<tr>
<th>Methods</th>
<th>Return Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>onclick(evt)</td>
<td>None</td>
<td>Handle click events.</td>
</tr>
<tr>
<td>onmouseover(evt)</td>
<td>None</td>
<td>Handle mouse over events.</td>
</tr>
<tr>
<td>onmouseout(evt)</td>
<td>None</td>
<td>Handle mouse out events.</td>
</tr>
<tr>
<td>getPosition()</td>
<td>Position</td>
<td>Returns the position for this dancer.</td>
</tr>
<tr>
<td>setPosition()</td>
<td>None</td>
<td>Sets the position for this dancer.</td>
</tr>
<tr>
<td>getOrientation()</td>
<td>Orientation</td>
<td>Returns the orientation for this dancer.</td>
</tr>
<tr>
<td>setOrientation()</td>
<td>None</td>
<td>Sets the orientation for this dancer.</td>
</tr>
<tr>
<td>setParent(group::Group)</td>
<td>None</td>
<td>Set the parent of this dancer to be the given group.</td>
</tr>
<tr>
<td>updateChildrenBasedOnMyShape(currentTime::Time)</td>
<td>None</td>
<td>Called by the group this dancer is a child of. By default this does nothing.</td>
</tr>
<tr>
<td>setMyPositionAndModifyChildren(position::Position)</td>
<td>None</td>
<td>Called by the group this dancer is a child of. Modifies the position of this dancer.</td>
</tr>
</tbody>
</table>

## Static Methods

<table>
<thead>
<tr>
<th>Static Methods</th>
<th>Return Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>updateTracking()</td>
<td>None</td>
<td>Update the side list of tracked dancer names.</td>
</tr>
</tbody>
</table>

## Properties

<table>
<thead>
<tr>
<th>Properties</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>dancerId</td>
<td>number</td>
<td>A unique integer to identify this dancer.</td>
</tr>
<tr>
<td>dancerTypeld</td>
<td>DancerTypeld</td>
<td>The dancer type of this dancer.</td>
</tr>
<tr>
<td>Property</td>
<td>Type</td>
<td>Description</td>
</tr>
<tr>
<td>-------------------</td>
<td>-----------------------</td>
<td>---------------------------------------------------------------</td>
</tr>
<tr>
<td>dancerShape</td>
<td>DancerShape</td>
<td>The shape that is displayed on screen to represent the dancer.</td>
</tr>
<tr>
<td>dancerSize</td>
<td>DancerShapeSize</td>
<td>The size of the dancer on screen.</td>
</tr>
<tr>
<td>dancerName</td>
<td>string</td>
<td>The name of this dancer.</td>
</tr>
<tr>
<td>parentGroup</td>
<td>Group</td>
<td>The group this dancer currently belongs to.</td>
</tr>
<tr>
<td>dancerColor</td>
<td>string</td>
<td>The color for the underlying element that is bound to this dancer.</td>
</tr>
<tr>
<td>position</td>
<td>Position</td>
<td>The current absolute position of the dancer.</td>
</tr>
<tr>
<td>orientation</td>
<td>Orientation</td>
<td>The current orientation of the dancer.</td>
</tr>
<tr>
<td>element</td>
<td>SVG element</td>
<td>HTML element</td>
</tr>
</tbody>
</table>
dancevis.TimeManager class

Constructor

<table>
<thead>
<tr>
<th>Constructor</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TimeManager()</td>
<td>Creates a new time manager.</td>
</tr>
</tbody>
</table>

Methods

<table>
<thead>
<tr>
<th>Methods</th>
<th>Return Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>timer(position: Position)</td>
<td>None</td>
<td>Create a timer on screen and place it at the given position.</td>
</tr>
<tr>
<td>scheduleGroup(group)</td>
<td>None</td>
<td>Schedule a group to receive updates.</td>
</tr>
<tr>
<td>annotateAt(position)</td>
<td>None</td>
<td>Tell the time manager where on screen to place annotations.</td>
</tr>
<tr>
<td>annotate(str, startTime, endTime)</td>
<td>None</td>
<td>Add an annotation to the pool of time dependent annotations.</td>
</tr>
<tr>
<td>onTimeStep()</td>
<td>None</td>
<td>Function that handles notification of the current time to all scheduled groups.</td>
</tr>
<tr>
<td>play()</td>
<td>None</td>
<td>Run the simulation.</td>
</tr>
<tr>
<td>pause()</td>
<td>None</td>
<td>Pause the simulation.</td>
</tr>
<tr>
<td>reset()</td>
<td>None</td>
<td>Reset the simulation to its starting point.</td>
</tr>
</tbody>
</table>

Static Methods

<table>
<thead>
<tr>
<th>Static Methods</th>
<th>Return Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>formatedTimeStr(currentTime: Time)</td>
<td>string</td>
<td>Return the time in a formatted string #:#.#.#.##</td>
</tr>
</tbody>
</table>

Properties

<table>
<thead>
<tr>
<th>Properties</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>numMillisecondsPerInterval</td>
<td>number</td>
<td>The number of milliseconds between updates.</td>
</tr>
<tr>
<td>groups</td>
<td>List.&lt;group&gt;</td>
<td>The list of groups that have been scheduled.</td>
</tr>
<tr>
<td>currentTime</td>
<td>Time</td>
<td>The current simulation time.</td>
</tr>
<tr>
<td>--------------</td>
<td>-------------------</td>
<td>------------------------------</td>
</tr>
<tr>
<td>annotations</td>
<td>List.&lt;Object&lt;time, time, string&gt;&gt;</td>
<td>A list of annotations.</td>
</tr>
</tbody>
</table>