MECHO 4.0 – КРАТЪК СПРАВОЧНИК

I. Включване и инициализиране на Mecho

Програмите на Mecho са на JavaScript в рамките на HTML5 страница. Библиогеката Mecho е във файла mecho.min.js. В HTML файла трябва да се включи библиотеката и след зареждането на страницата - да се инициализира.

I.1. Включване

Включването на библиотеката е с тага <script> в частта <head>:

```
<head>
   <script src="mecho.min.js"></script>
</head>
```

I.2. Инициализиране

Инициализиране на библиотеката след зареждането на страницата става като в атрибута onload на <body> се посочва функция, в която се създава инстанция на Mecho. Функцията е добре да е в <script> в <head>. Името на функцията не е фиксирано, но по традиция се използва main.

```
<head>
   <script>
      function main()
      {
         new Mecho();
      }
   </script>
</head>

<body onload="main()">
   :
</body>
```

При правилно включване и инициализиране на Mecho трябва да се появи графично поле с равнина на квадратчета.

II. Настройка на сцената

II.1. Реферирание

За да се настрои сцената (т.е. виртуалната среда, графичното поле) тя трябва да бъде запазена в променлива.:

```
сцена = new Mecho();
```

II.2. Небе

Небето е общия фон на сцената. То се характеризира единствено със своя цвят. Цветът на небето/фона се задава с вграден или с потребителски цвят на *сцена*.sky. По подразбиране цветът на небето е бял Mecho.WHITE.

a) Вграден цвят

За вграден цвят се използва:

```
сцена.sky = Mecho.цвят;
```

като *цвят* е BLACK, WHITE, YELLOW, BLUE, RED или GREEN.

b) Потребителски цвят

Цвят на небето чрез потребителски цвят:

```
сцена.sky = [червено, зелено, синьо];
```

като *червено*, *зелено* и *синьо* са цветовите компоненти на RGB цвят и всяко от тях е число от 0 до 1.

II.3. Земя

Земята е равнина плоскост, която освен цвят, може да има и материя. По подразбиране земята е направена от материала плочки Mecho.TILE.

a) Цвят на земята

Цветът на земята се задава подобно на цвета на небето, но се записва в *сцена*.ground.

b) Материал на земята

В допълнение земята може да е от даден материал:

```
сцена.ground = Mecho.материал;
```

като *материал* е TILE, WOOD, GOLD, METAL, SCRATCH, METRIC, PAPER, ASPHALT, MARBLE, WATER, ROCK, ROCK2 или INDUSTRIAL.

c) Скриване на земята

Земята може да бъде скрита като вместо цвят или материал се подаде false:

```
сцена.ground = false;
```

Обратното показване на земята е като се подаде true.

II.4. Бутони

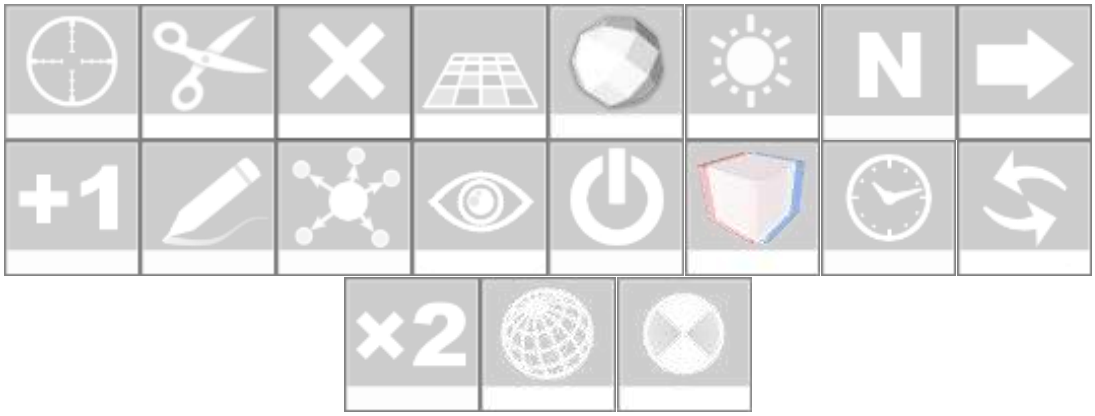
Бутони се създават с функцията button и се разполагат отгоре надолу в горния ляв ъгъл на сцената. Бутоните се характеризират с картинка, клавиш за активиране и функция, която реализира действието им.

a) Обикновени бутони

Нямат нужда от запазването им в променлива, понеже не връщат никаква обратна информация. Създават се с:

```
button('картинка', 'клавиш', функция);
```

където *картинка* е стринг с името на картинка за бутон от папката images/buttons и е една от следните стойности center, cut, exit, ground, half, light, n, next, one, pencil, random, show, start, stereo, time, toggle, two, wireframe или xray.



Клавиш е стринг с името на клавиш, натискането на който активира бутона (каквото би станало и с кликване с мишката върху бутона). Допустими са следните стойности за *клавиш*: CANCEL, HELP, BACK_SPACE, TAB, CLEAR, RETURN, ENTER, SHIFT, CONTROL, ALT, PAUSE, CAPS_LOCK, ESCAPE, SPACE, PAGE_UP, PAGE_DOWN, END, HOME, LEFT, UP, RIGHT, DOWN, PRINTSCREEN, INSERT, DELETE, 0 до 9, SEMICOLON, EQUALS, A до Z, CONTEXT_MENU, NUMPAD0 до NUMPAD9, MULTIPLY, ADD, SEPARATOR, SUBTRACT, DECIMAL, DIVIDE, F1 до F24, NUM_LOCK, SCROLL_LOCK, COMMA, PERIOD, SLASH, BACK_QUOTE, OPEN_BRACKET, BACK_SLASH, CLOSE_BRACKET, QUOTE и META.

Функция е име на потребителска функция на JavaScript, подадена без (…). Тази функция се извиква при кликване върху бутона или натискане на клавиша за бутона.

Пример за създаване на бутон с картинка images/buttons/random.png, който може да се активира с клавиша интервал/шпация и при активирането му се изпълнява функцията myAction:

```
button('random','space',myAction);
:
function myAction()
{
   :
}
```

Като функция може да се подаде и анонимна функция:

```
button('random','space',function (){...});
```

b) Бутони със състояния

Тези бутони имат определен брой състояния, обозначени с цели числа от 0 нагоре. Визуално тези състояния се представят като малки квадратчета вдясно от бутона. Текущото състояние е отбелязано със запълнено квадратче.

Създаването на такъв бутон става с:

```
бутон = button('картинка', 'клавиш', функция, състояния, начално);
```

където *картинка*, *клавиш* и *функция* са същите като при обикновените бутони, *състояния* е цяло число от 1 нагоре за общия брой състояние, а *начало* е номерът на началното състояние – число от 0 до *състояния*-1 (ако този параметър липсва, се приема, че е 0).

c) Използване на бутон със състояния

Активирането на бутон със състояния автоматично прехвърля бутона в следващото състояние преди извикването на потребителската функция. Тя може да използва свойството this.state, за достъп до текущото състояние:

```
b = button('random','space',myAction,3);
:
function myAction()
{
   if (this.state==1) {...}
}
```

III. Графични обекти

В библиотеката Mecho са реализирани различни графични обекти. Те са групирани в три групи. Имат някои свойства, които са общи за всички, но имат и индивидуални свойства.

III.1. Геометрични обекти

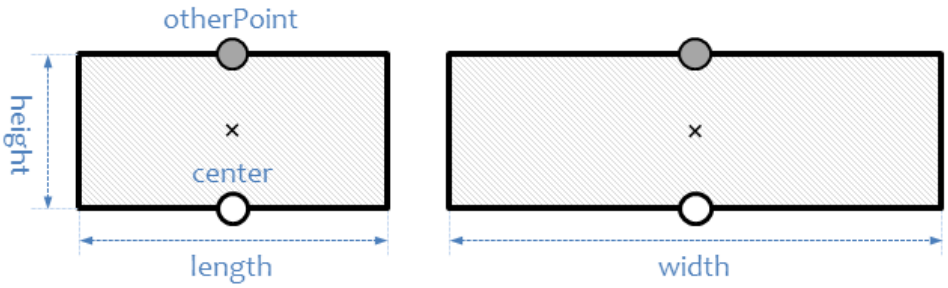
В тази група са включени графичните обекти, които са построени на базата на традиционни геометрични обекти – куб, правоъгълен паралелепипед, сфера, елипсоид, цилиндър.

а) Кутия (box)

Създаване на кутия става с командите:

```
box(center)
box(center,length)
box(center,length,width)
box(center,length,width,height)
```

където center е центърът на куба (масив от 3 числа), length е дължина, width е широчина, а height е височина.



Ако е зададена само дължина, създадената кутия е с формата на куб. Също куб е и ако трите размера са равни.

Обектът box поддържа следните общи свойства:

- center – ефективен център на кутията
- centerOffset – отместване на центъра спрямо вградения център (по подразбиране е [0,0,-height/2], т.е. центърът е преместен в средата на долната стена)
- imageOffset – отместване на образа на обекта (по подразбиране е undefined)
- parent – родителски елемент (по подразбиране е undefined)
- material – цвят или материал на кутията
- tiles – мащаб на шарката на кутията (масив от 3 числа за мащаби по X, Y и Z)
- visible – видимост на кутията
- otherPoint – средата на горната стена (по локалната Z ос)
- atPoint – относителна точка спрямо вградения център

Обектът box поддържа следните специфични свойства и методи:

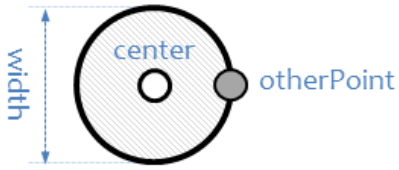
- length – дължина - размер по оста X (по подразбиране е 1)
- width – широчина - размер по оста Y (по подразбиране е length)
- height – височина - размер по оста Z (по подразбиране е width)

b) Топка (ball)

Създаване на топка става с командите:

```
ball(center)
ball(center,width)
```

където center е центърът на топката (масив от 3 числа), width е диаметърът.



Обектът ball поддържа следните общи свойства:

- center – център на топката
- centerOffset – отместване на центъра (по подразбиране е [0,0,0])
- imageOffset – отместване на образа на обекта (по подразбиране е undefined)
- parent – родителски елемент (по подразбиране е undefined)
- material – цвят или материал на топката
- tiles – мащаб на шарката на кутията (масив от 2 числа за хоризонтален и вертикален мащаб)
- visible – видимост на топката
- nice – красота на топката (по подразбиране е true)
- otherPoint – точка по „екватора“ на топката (по локалната X ос)
- atPoint – относителна точка спрямо вградения център

Обектът ball поддържа следните специфични свойства:

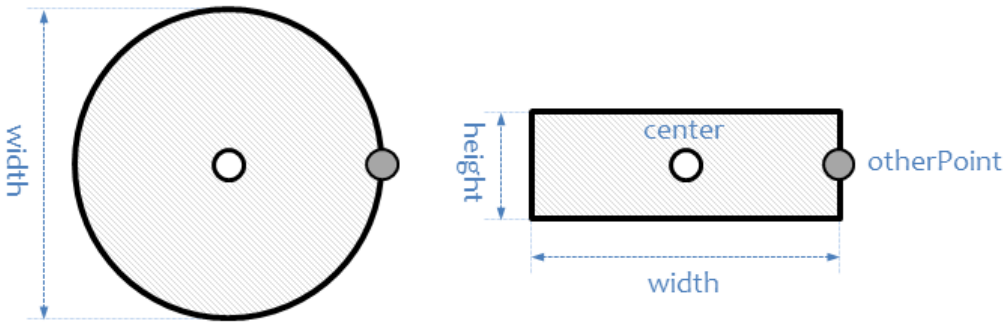
- width – широчина/диаметър (по подразбиране е 1)

c) Диск (disc)

Създаване на диск става с командите:

```
disk(center)
disk(center,width)
disk(center,width,height)
```

където center е центърът на диска (масив от 3 числа), width е диаметърът, а height е височината.



Обектът disk поддържа следните общи свойства:

- center – център на диска
- centerOffset – отместване на центъра, по подразбиране е [0,0,0]
- imageOffset – отместване на образа на обекта (по подразбиране е undefined)
- parent – родителски елемент (по подразбиране е undefined)
- material – цвят или материал на диска, допустими са два материала – първият за околната стена, вторият за двете основи
- tiles – мащаб на шарката на кутията (масив от 4 числа – първите две са за хоризонтален и вертикален мащаб на материала на околната стена, вторите две са мащабите по X и Y на материала на двете основи)
- visible – видимост на диска
- nice – красота на диска (по подразбиране е true)
- hollow – кухост на диска (при true не се рисуват двете основи)
- otherPoint – точка по „периферията“ на диска (по локалната X ос)
- atPoint – относителна точка спрямо вградения център

Обектът disk поддържа следните специфични свойства:

- width – широчина/диаметър (по подразбиране е 2)
- height – височина (по подразбиране е 1)

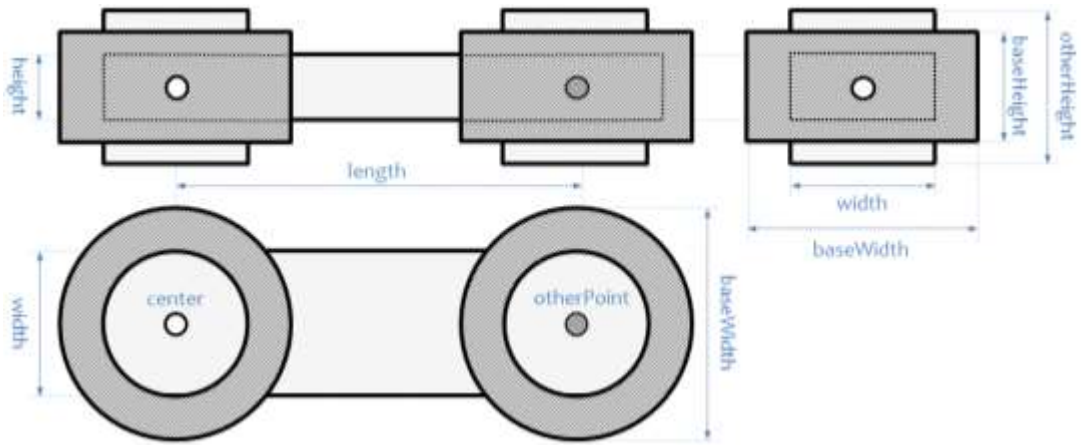
III.2. Инженерни обекти

а) Греда (beam)

Създаване на греда става с командата:

```
beam(center)
beam(center,length)
beam(center,length,width)
beam(center,length,width,height)
beam(center,length,width,height,baseWidth)
beam(center,length,width,height,baseWidth,baseHeight)
beam(center,length,width,height,baseWidth,baseHeight,otherHeight)
```

където center е центърът на гредата (масив от 3 числа), width е широчината, а height е височината. В двата края има дискове от друг материал, които са с широчина baseWidth и височина baseHeight. В двата края има и ос с височина otherHeight и широчина, съвпадаща с width.



Дисковете се рисуват само ако baseWidth>width и baseHeight>height.
Допълнителните оси - само ако otherHeight>height и otherHeight>baseHeight.

Обектът beam поддържа следните общи свойства:

- center – център на гредата
- centerOffset – отместване на центъра, по подразбиране е [0,0,0]
- imageOffset – отместване на образа на обекта (по подразбиране е undefined)
- parent – родителски елемент (по подразбиране е undefined)
- material – цвят или материал на гредата, допустими са два материала – първият за самата греда, вторият е за двата диска
- tiles – мащаб на шарката на гредата (масив от 11 числа – първите три са за самата греда, две са за диска при центъра, две са за издадъка при центъра, две за диска при другата точка и последните две за издадъка при нея
- visible – видимост на гредата
- nice – красота на диска (по подразбиране е true)
- otherPoint – точка в другия край на гредата (по локалната X ос)
- atPoint – относителна точка спрямо вградения център

Обектът beam поддържа следните специфични свойства:

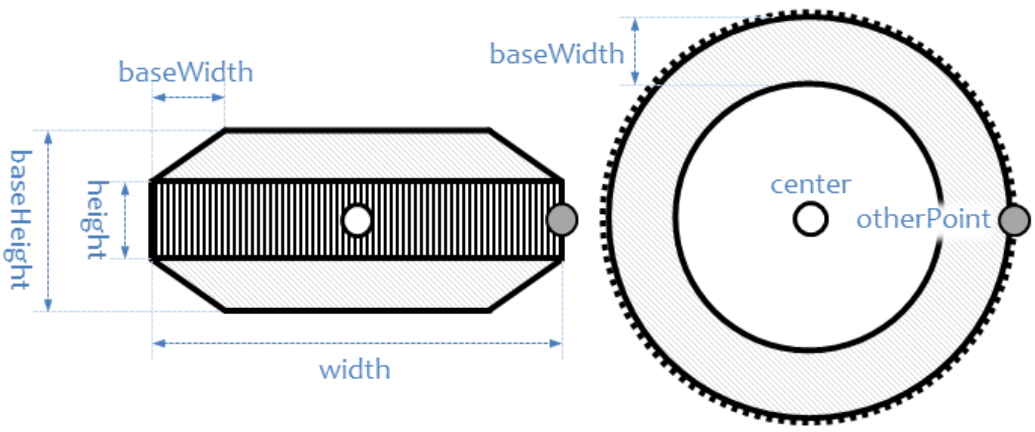
- length – дължина (размер по оста X, по подразбиране е 10)
- width – широчина (размер по оста Y, по подразбиране е 1)
- height – височина (размер по оста Z, по подразбиране е 0.25)
- baseWidth – широчина/диаметър на дисковете в двата края (по подразбиране е 2*width)
- baseHeight – височина на дисковете в двата края (по подразбиране е 2*height)
- otherHeight – височина на допълнителните оси (по подразбиране е baseHeight+height)

b) Зъбно колело (gear)

Създаване на зъбно колело със зъбци по външния периметър става с командата:

```
gear(center)
gear(center,width)
gear(center,width,height)
gear(center,width,height,baseWidth)
gear(center,width,height,baseWidth,baseHeight)
```

където center е центърът на зъбното колело (масив от 3 числа), width е широчината, height е височината, baseWidth е широчината на периферията, а baseHeight е височината ѝ.



Обектът gear поддържа следните общи свойства:

- center – център на зъбното колело
- centerOffset – отместване на центъра, по подразбиране е [0,0,0]
- imageOffset – отместване на образа на обекта (по подразбиране е undefined)
- parent – родителски елемент (по подразбиране е undefined)
- material – цвят или материал на зъбното колело
- tiles – мащаб на шарката (масив от 4 числа – хоризонтален мащаб на външната и вътрешната стена, вертикален мащаб на външната стена, вертикален мащаб на вътрешната стена и хоризонтален мащаб на скосените стени)
- visible – видимост на зъбното колело
- nice – красота на зъбното колело (по подразбиране е true)
- otherPoint – точка в периферията на зъбното колело (по локалната X ос)
- atPoint – относителна точка спрямо вградения център

Обектът gear поддържа следните специфични свойства:

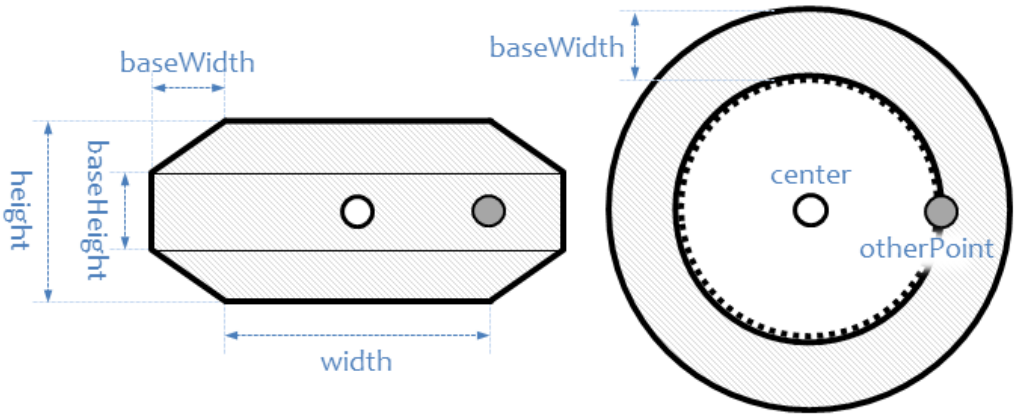
- width – широчина (външен диаметър, по подразбиране е 5)
- height – височина (по подразбиране е 1)
- baseWidth – дебелина на зъбното колело (по подразбиране е 1, ако width≥4, иначе е width/4)
- baseHeight – височина на вътрешния диаметър (по подразбиране е 2*height)
- gears – коефициент за гъстота за зъбците (по подразбиране е 1)

c) Зъбен пръстен (ring)

Създаване на зъбен пръстен със зъбци по вътрешния периметър става с командата:

```
ring(center)
ring(center,width)
ring(center,width,height)
ring(center,width,height,baseWidth)
ring(center,width,height,baseWidth,baseHeight)
```

където center е центърът на зъбния пръстен (масив от 3 числа), width е широчината, height е височината, baseWidth е широчината на периферията, а baseHeight е височината ѝ.



Обектът ring поддържа следните общи свойства:

- center – център на зъбния пръстен
- centerOffset – отместване на центъра, по подразбиране е [0,0,0]
- imageOffset – отместване на образа на обекта (по подразбиране е undefined)
- parent – родителски елемент (по подразбиране е undefined)
- material – цвят или материал на зъбния пръстен
- tiles – мащаб на шарката (масив от 4 числа – хоризонтален мащаб на външната и вътрешната стена, вертикален мащаб на външната стена, вертикален мащаб на вътрешната стена и хоризонтален мащаб на скосените стени)
- visible – видимост на зъбния пръстен
- nice – красота на зъбния пръстен (по подразбиране е Mecho.VERYTRUE)
- otherPoint – точка по вътрешната периферия на зъбния пръстен (по локалната X ос)
- atPoint – относителна точка спрямо вградения център

Обектът ring поддържа следните специфични свойства:

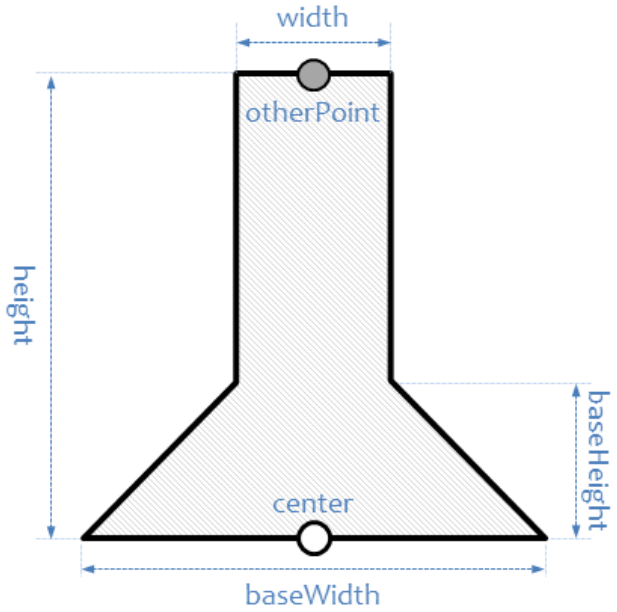
- width – широчина (вътрешен диаметър, по подразбиране е 5)
- height – височина (по подразбиране е 2)
- baseWidth – дебелина на зъбния пръстен (по подразбиране е 1)
- baseHeight – височина на външния диаметър (по подразбиране е height/2)
- gears – коефициент за гъстота за зъбците (по подразбиране е 1)

d) Пилон (pillar)

Създаване на пилон става с командата:

```
pillar(center)
pillar(center,height)
pillar(center,height,width)
pillar(center,height,width,baseHeight)
pillar(center,height,width,baseHeight,baseWidth)
```

където center е центърът на пилона (масив от 3 числа), height е височината, radius е радиусът в горния край, heightBase е височината на основата, а radiusBase е радиусът на основата.



Обектът pillar поддържа следните общи свойства:

- center – център на пилона в центъра на долната основа
- centerOffset – отместване на центъра, по подразбиране е [0,0,0]
- imageOffset – отместване на образа на обекта (по подразбиране е undefined)
- parent – родителски елемент (по подразбиране е undefined)
- material – цвят или материал на пилона, допустими са два материала – първият за околната стена, вторият за горната основи
- tiles – мащаб на шарката на пилона (масив от 4 числа – първите две са за хоризонтален и вертикален мащаб на материала на околната стена, вторите две са мащабите по X и Y на материала на горната основа)
- hollow – кухост, отнася се за долната основа, по подразбиране е true и долната основа не се рисува, горната основа винаги се рисува
- visible – видимост на пилона
- otherPoint – точка в центъра на горната основа на пилона (по локалната Z ос)
- atPoint – относителна точка спрямо вградения център

Обектът pillar поддържа следните специфични свойства:

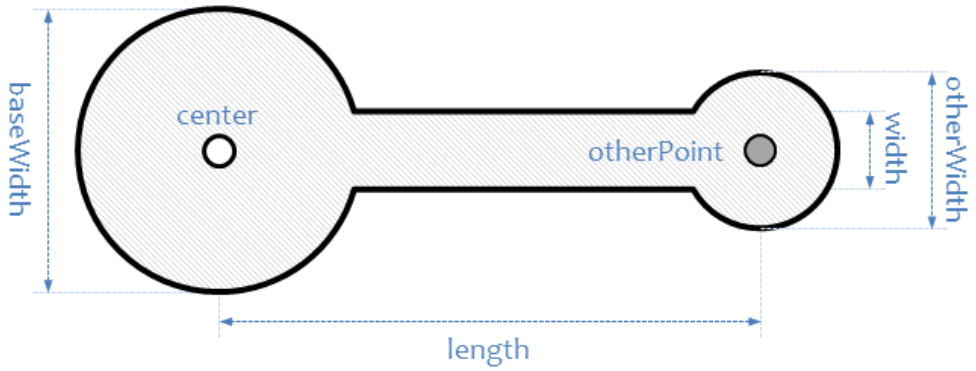
- height – височина на пилона (по подразбиране е 10)
- width – широчина на основната част (по подразбиране е 1)
- baseHeight – височина на подпората (по подразбиране е 1, ако height≥2, иначе е height/2)
- baseWidth – широчина на подпората (по подразбиране е width+2*baseHeight)

е) Релса (rail)

Създаване на релса става с командите:

```
rail(center)
rail(center,length)
rail(center,length,width)
rail(center,length,width,baseWidth)
rail(center,length,width,baseWidth,otherWidth)
```

където center е центърът на релсата (масив от 3 числа) – това е единият ѝ край, length е дължината на релсата, width е широчината, а baseWidth и otherWidth са широчините на ограничителите в двата края. Ако техните размери са по-малки от width, тогава не се рисуват.



Обектът rail поддържа следните общи свойства:

- center – център на релсата в единия от краищата ѝ
- centerOffset – отместване на центъра, по подразбиране е [0,0,0]
- imageOffset – отместване на образа на обекта (по подразбиране е undefined)
- parent – родителски елемент (по подразбиране е undefined)
- material – цвят или материал на релсата
- tiles – мащаб на шарката на релсата (масив от 6 числа – първите две са за самата релса, вторите две са мащабите по X и Y на материала на ограничителната сфера при центъра, а последните две са мащабите на шарката на другата ограничителна сфера)
- visible – видимост на релсата
- nice – красота на релсата и ограничителите
- otherPoint – точка в другия край на релсата (по локалната Z ос)
- atPoint – относителна точка спрямо вградения център

Обектът rail поддържа следните специфични свойства:

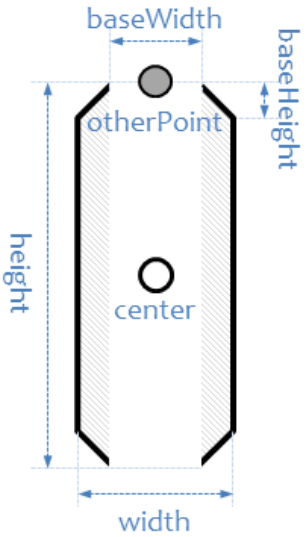
- length – дължина на релсата (по подразбиране е 10)
- width – широчина на релсата (по подразбиране е 0.3)
- baseWidth – размер на ограничителната сфера при центъра (по подразбиране е 2*width)
- otherWidth – размер на другата ограничителна сфера (по подразбиране е baseWidth)

f) Тръба (tube)

Създаване на тръба става с командите:

```
tube(center)
tube(center,height)
tube(center,height,width)
tube(center,height,width,baseHeight)
tube(center,height,width,baseHeight,baseWidth)
```

където center е центърът на тръбата (масив от 3 числа), height е височината, width е широчината, baseHeight е височината на скосените части, baseWidth е широчината на отворите. Ако baseWidth>width скосените части са обърнати навън и отворът е по-широк от самата тръба.



Обектът tube поддържа следните общи свойства:

- center – център на тръбата
- centerOffset – отместване на центъра, по подразбиране е [0,0,0]
- imageOffset – отместване на образа на обекта (по подразбиране е undefined)
- parent – родителски елемент (по подразбиране е undefined)
- material – цвят или материал на тръбата
- tiles – мащаб на шарката на тръбата (масив от 4 числа – първите две са за самата тръба, вторите две са скосените зони)
- visible – видимост на тръбата
- nice – красота на тръбата и скосените зони
- hollow – кухост на тръбата (при true не се рисува скосената зона в противоположния край спрямо otherPoint)
- otherPoint – точка в другия край на тръбата (по локалната Z ос)
- atPoint – относителна точка спрямо вградения център

Обектът tube поддържа следните специфични свойства:

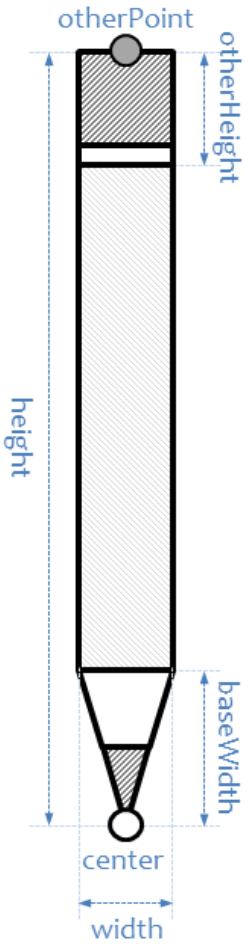
- height – височина на тръбата (по подразбиране е 6)
- width – широчина на релсата (по подразбиране е 0.3)
- baseWidth – размер на ограничителната сфера при центъра (по подразбиране е 2*width)
- otherWidth – размер на другата ограничителна сфера (по подразбиране е baseWidth)

g) Молив (pencil)

Създаване на молив става с командите:

```
pencil(center)
pencil(center,height)
pencil(center,height,width)
pencil(center,height,width,baseHeight)
pencil(center,height,width,baseHeight,otherHeight)
```

където center е центърът на тръбата (масив от 3 числа), height е височината, width е широчината, baseHeight е височината на подострената част, а otherHeight е височината на гумата.



Обектът pencil поддържа следните общи свойства:

- center – център на молива във върха му
- centerOffset – отместване на центъра, по подразбиране е [0,0,0]
- imageOffset – отместване на образа на обекта (по подразбиране е undefined)
- parent – родителски елемент (по подразбиране е undefined)
- material – цвят (но не и материал) на молива
- visible – видимост на молива, не се отнася за нарисуваната линия
- hollow – кухост на молива (при true не се рисува гумата)
- otherPoint – точка в края на гумата на молива (по локалната Z ос)
- atPoint – относителна точка спрямо вградения център

Обектът pencil поддържа следните специфични свойства:

- height – височина на молива (по подразбиране е 10)
- width – широчина на молива (по подразбиране е 0.6)
- baseHeight – размер на подостряната част (по подразбиране е 1.5*width)
- otherHeight – размер на гумата (по подразбиране е 1)

Свойства за рисуване:

- down – ако стойността е true, моливът започва да рисува там, където е центъра му; ако стойността е число, моливът почва да рисува за определено от числото време (в секунди) и после автоматично спира да рисува; ако стойността е false, моливът спира да рисува (по подразбиране е false)
- up – ако стойността е true, моливът спира да рисува; ако стойността е false, започва да рисува (по подразбиране е true)
- downNext – ако стойността е true, рисуването започва след първото преместване на молива, т.е. текущата позиция на молива не участва в рисуваната следа; ако стойността е false, рисуването започва от текущата позиция (по подразбиране е true)

III.3. Специални обекти

a) Гледна точка

Обектът определя къде е „камерата“, която определя коя част от сцената се вижда на екрана. Обектът се създава автоматично за всяка сцена и достъпът до него е чрез сцена.viewObject.

Гледната точка има следните свойства:

- target – тримерна точка в пространството, към която е обърната камерата, тази точка се „вижда“ в средата на графичното поле
- eye – тримерна точка, в която се намира камерата, т.е. сцената се показва като видяна от тази точка
- up – вектор, който указва посоката „нагоре“ спрямо гледащия (по подразбиране е [0,0,1])
- distance – разстояние между точката, от която се гледа (eye), и точката, към която се гледа (target), промяната на свойството запазва target, но преизчислява eye
- alpha – ъгъл на хоризонтално завъртане на точката, от която се гледа (eye), около точката, към която се гледа (target), промяната на свойството запазва target, но преизчислява eye
- beta – ъгъл на вертикално завъртане на точката, от която се гледа (eye), около точката, към която се гледа (target), промяната на свойството запазва target, но преизчислява eye
- follow – обект за следене, ако стойността е валиден графичен обект, то target плавно се измества към центъра на обекта

b) Цел

Целевият обект е помощен графичен обект, който показва къде се намира точката, към която гледаме и спрямо която се върти сцената при навигация. В допълнение, обектът показва посоките X и Y.

Обектът се създава с командата:

```
сцена.targetObject = new Mecho.Target();
```

където *сцена* е създадената с new Mecho сцена.

Обектът трябва да има специалното име targetObject. Той автоматично се прави видим при започване на навигация (с мишка или клавиатура) и след приключване отново се прави невидим.

IV. Свойства

IV.1. Разположение

Всеки обект в своята вградена дефиниция има основна точка, наречена център. Той може да съвпада с геометричния център на обекта, но това не е задължително. Центърът на обект служи за две основни дейности:

- Преместване на обект, чрез промяна на центъра му
- Завъртане на обект около центъра му

За практически цели е възможно да се избере друга точка от обект да бъде център. Това става с отместване на центъра спрямо вграденото му положение.

a) Център (center)

Свойство `center`, което отговаря на центъра на обекта и е фиксирана точка спрямо него. Центърът е масив от 3 числа – координатите на обекта.

```
обект.center = [x,y,z];
```

Свойството center е вградения център на обекта отместен с centerOffset.

b) Отместване на центъра (centerOffset)

Свойството `centerOffset` показва коя точка спрямо вградения център на обекта да се използва като негов център. Стойността на свойството е вектор, който се прибавя към вградения център на обекта за получаване на ефективния център на обекта.

Пример с куб със страна 10. Следната команда определя ефективният център на куба да е геометричния център:

```
куб.centerOffset = [0,0,0];
```

А следната команда определя ефективния център да е в средата на долната стена:

```
куб.centerOffset = [0,0,-5];
```

c) Другата точка (otherPoint)

Всеки обект има втора по важност точка, която се използва за свързване с други обекти. Тази точка най-често се намира „в другия край“ на обекта или някъде по периферията му. Тази точка е дефинирана специфично за всеки обект.

Свойството otherPoint е само за четене, то не може да бъде променяно.

```
обект1.center = обект2.otherPoint;
```

Стойността на свойството otherPoint съвпада със стойността на atPoint(1).

d) Точка от обекта (atPoint)

Методът atPoint се използва за намиране на координатите на точка спрямо локалната координатна система на обект – т.е. отчитайки неговото положение и ориентация в пространството. Параметрите на atPoint са относителни числа.

Методът е дефиниран в кратка и пълна форма. Кратката форма е:

```
обект.atPoint(relPos);
```

където `relPos` е дробно число, указващо относителна позиция. Кратката форма връща координати по правата, която минава през вградения център на обекта (т.е. този center, който е при centerOffset=[0,0,0]) и през другата му точка (т.е. otherPoint). Конкретната стойност на relPos има следния смисъл:

- Вграденият център на обект съответства на atPoint(0)
- Другата точка otherPoint на обект съответства на atPoint(1)

При relPos>1 се изчисляват точки отвъд другата точка, а при relPos<0 – точки в обратната посока.

Пълната форма е:

```
обект.atPoint(relX,relY,relZ);
```

където relX, relY и relZ са дробни число, указващи относителна позиция по всяка от осите спрямо вградения център на обект. Стойности 1 отговарят на точки по или около периферията на обекта в даденото направление.

За всеки обект една от локалните оси се приема за главна ос. Кратката форма на atPoint е с параметър по тази ос. Например, за обекта box главната локална ос е Z, затова стойността на atPoint(1) е същата като стойността на atPoint(0,0,1).

e) Отместване на образа (imageOffset)

Свойството imageOffset е вектор, с който се премества образа на обекта. Векторът е спрямо локалната координатна система на обекта. Центърът на обекта center не се променя, не се променят otherPoint и atPoint.

f) Родителски обект (parent)

Свойството parent е друг графичен обект, чиято ориентация и позиция в пространството се приема за глобална за обекта. По този начин центъра на обекта ще е спрямо координатната система на родителския обект. Координатните оси ще са спрямо локалните оси на родителския обект.

Свойството parent се използва за свързване на обекти към други обекти, например, създаване на модел на скелет. Със свойството се прави възможна каскадната ориентация на обектите – т.е. промяната на ориентацията на обект да се простира автоматично към всички свързани с него обекти.

IV.2. Цвят и материал

Всеки обект има материал, от който е направен. Този материал оказва влияние единствено на изобразяването на обекта. Свойството е material.

a) Цвят (material)

За цвят може да се използва както някой от вградените цветове, така и произволен потребителски цвят:

```
обект.material = Mecho.цвят;
обект.material = [червено, зелено, синьо];
```

като *цвят* е BLACK, WHITE, YELLOW, BLUE, RED или GREEN; *червено*, *зелено* и *синьо* са цветовите компоненти на RGB цвят и всяко от тях е число от 0 до 1.

b) Материал (material)

За материал може да се използва някой от вградените материали:

```
обект.material = Mecho.материал;
```

като *материал* е един от материалите: TILE, WOOD, WOOD_ROUND, DARK_WOOD, DARK_WOOD_ROUND, GOLD, METAL, SCRATCH, METRIC, PAPER, ASPHALT, MARBLE, WATER, ROCK, ROCK2 или INDUSTRIAL.

Някои обекти могат да използват няколко материала. За тях стойността на material е масив от материали:

```
обект.material = [Mecho.материал1, Mecho.материал2, ...];
```

Допустими са и следните две алтернативни форми:

```
обект.material = true;
обект.material = false;
```

при които свойството material действа като visible, т.е. обектът става видим или невидим.

c) Размер на шарката (tiles)

При използването на материал се поставя шарка върху повърхността на обекта. Размерът (мащабът) на тази шарка се определя от свойството tiles. Стойността е масив от 2 или повече числа. Конкретната интерпретация на тези числа зависи от обекта.

```
обект.tiles = [коефициент1, коефициент2, ...];
```

като *коефициент1, коефициент2, ...* са числа.

d) Видимост (visible)

Свойството visible определя дали обект е видим или невидим:

```
обект.visible = видимост;
```

като *видимост* е true или false. Всеки обект се създава по подразбиране видим.

e) Кух обект (hollow)

Някои обекти позволяват да не се рисуват всичките им елементи (например, на диск да не се рисуват двете основи). За други обекти свойството hollow не оказва влияние.

Свойството hollow определя дали обект ще се рисува цял.

```
обект.hollow = кухост;
```

като *кухост* е true или false. Всеки обект се създава по подразбиране да не е кух.

Свойството се използва или за обекти, през които трябва да се вижда, или за обекти, на които елементите няма нужда да се рисуват, понеже са скрити в други елементи.

f) Красота (nice)

Свойството nice определя колко красиво се рисуват елементите на обект. Това оказва влияние на броя на фрагментите при закръглени повърхнини.

```
обект.nice = красота;
```

като *красота* е true или false.

Ако стойността на nice е true, закръглените елементи се рисуват с повече фрагменти и изглеждат по-гладки. В противен случай се използват по-малко на брой фрагменти и обектите изглеждат по-ръбести.

IV.3. Размери

a) Дължина (length)

Свойството length определя дължината на обект:

```
обект.length = размер;
```

като *размер* е реално число. Не всеки обект има свойство за дължина.

b) Широчина (width, baseWidth, otherWidth)

Свойствата width, baseWidth и otherWidth определят широчините на обект (или диаметъра на заоблен обект):

```
обект.width = размер;
обект.baseWidth = размер;
обект.otherWidth = размер;
```

като *размер* е реално число. Не всеки обект има свойство за широчина.

c) Височина (height, baseHeight и otherHeight)

Свойствата height, baseHeight и otherHeight определят височините на обект:

```
обект.height = размер;
обект.baseHeight = размер;
обект.otherHeight = размер;
```

като *размер* е реално число. Не всеки обект има свойство за височина. Някои обекти имат по две височини.

V. Други

V.1. Промени по Array

Стандартният обект Array е с разширен прототип. Включени са нови свойства. Те се използват при работа с координати и вектори.

Първите три индекса на всеки масив могат да се адресират поименно с x, y и z, като:

масив.x ≡ *масив*[0]
масив.y ≡ *масив*[1]
масив.z ≡ *масив*[2]

Свойствата са достъпни както за четене, така и за промяна.