OOP Mine Sweeper Documentation

GitLab: https://gitlab.fit.cvut.cz/kachaana/OOP

Members:
Boiko Mykyta (@boikomyk)
Vladyslav Zavirskyy (@zavirvla)
Grankin Daniil (@grankdan)
Kachan Anastasiya (@kachaana)

Frameworks and Libraries

Because our game is build on Bloc the framework should be downloaded to run our code. The following installs #core version of Brick and Bloc without tests, development tools and experimental features.

```
Metacello new
    baseline: 'Brick';
    repository:
'github://pharo-graphics/Brick/src';
    load: #core
```

Instalation

The repository of the game is placed on the GitLab. There you can download project's packages: https://gitlab.fit.cvut.cz/kachaana/OOP
After downloading place them into Pharo's projects folder or just download it with Iceberg tool.
The game will be start after executing the following code in Playground:

```
MGameMenu open.
```

Description

The file structure of the game consists of the main package with execution code, tests package and resource package with icons which are used for design.

By default the game has following parameters: width = 10, height = 10, count of mines = 20. After game start's you can change them manually in settings.

Mouse Click's Legend

clickEvent - usual mouse left click for **openning** cell.
scrollEvent - scroll to **mark** cell.
dragEvent - drag cell to **unmark** it.

Using concepts and patterns

Singleton

The game is implemented with singleton pattern.

Code example:

```
MGameModel class
    instanceVariableNames: 'uniqueInstance'
```

```
MGameModel class>>getUniqueInstance
    ^ uniqueInstance ifNil:
 [ uniqueInstance := self withHeight: 10
width: 10 mines: 20 ]
```

Composite

The game window is composed of three elements: game grid with cells, which you can open, mark etc., timer which shows you time spent by playing and counter which count cells you've marked.

```
MGameStart class >> openInWindow: anElement
    | length |
    timerON := false.
    length:= (self size).
    (length < 180) ifTrue: [ length:= 180 ].
    font:= BlElement new size: (length)@(self size + 30);
background: Color black.
    timer:= MTimer new.
    counter:=MMinesLeftCounter new.
    counter getEl: anElement.
    space  := BlSpace new
          extent: (length)@(self size + 30);
          title: 'MineSweeper'.
    space root addChild: font; addChild: anElement;
addChild: timer; addChild: (counter relocate: (length -
55)@0);
          yourself.
       space show.
```

Hook & Template

Because we're working with graphics in Pharo, some concepts have already existed and we used them. For example it will be "drawOnSpartaCanvas:" method of class **BlElement**.

```
MMinesLeftCounter >>
drawOnSpartaCanvas: aCanvas
    |font|
     font := aCanvas font
     named: 'Source Sans Pro';
          size: 26;
          build.
    self size: 50 @ 30.
 aCanvas text
          baseline: (10@24);
          font: font;
          paint: Color white;
          string: self count asString;
          draw.

    ^self.
```

Observer

We use announcements which is Pharo implementation of observer pattern. In our project announcements are used to notify events such as opening or marking cell.