Aluno: Guilherme Marcello Casagranda

Compreensão de Programas

Breakout

1) Ferramenta utilizada: Plato

2) Arquivos verificados:
    - *'main.js',*
    - *'renderer.js',*
    - *'components/Controls.js',*
    - *'components/Draw.js',*
    - *'components/Game.js'*

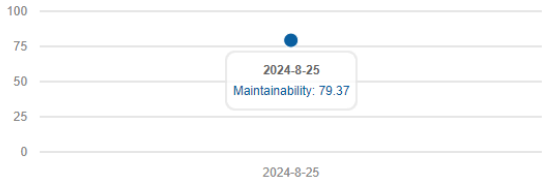
3) Métricas:
    - *Maintainability:* Um valor de 0 a 100 que representa a dificuldade relativa de se manter um código. Quanto maior melhor.
    - *Lines of code:* Número de linhas de código do arquivo.
    - *Difficulty:* Dificuldade relativa de escrever e entender o programa.
    - *Estimated errors:* número estimado de erros pela métrica de Halstead.
    - *Function weight:* Demonstra a complexidade (caminhos distintos possíveis em um bloco de código) e número de linhas de código de cada função.

4) Resultados:
   - *main.js:*

main.js

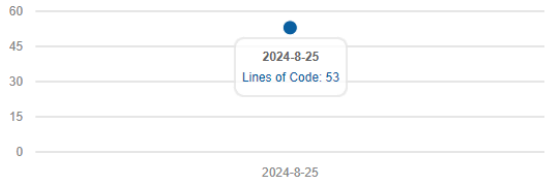
Maintainability

79.37



Lines of code

53



Difficulty

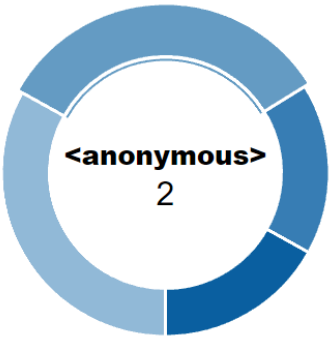
10.40

Estimated Errors

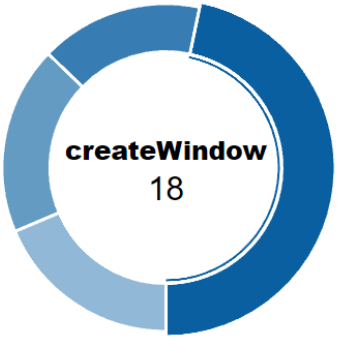
0.17

Function weight

By Complexity



By SLOC

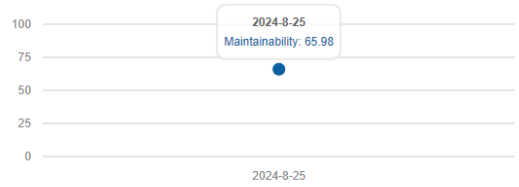


- *'renderer.js':*

renderer.js

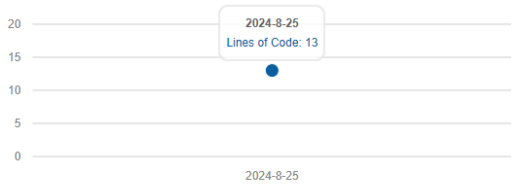
Maintainability

65.98



Lines of code

13



Difficulty

4.29

Estimated Errors

0.10

Function weight

By Complexity

By SLOC

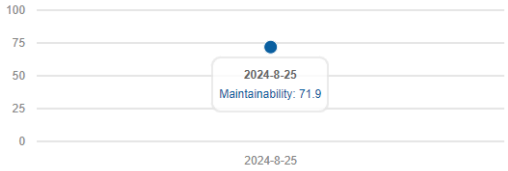
```
1  const canvas = document.getElementById("myCanvas");
   Column: 1 "'const' is available in ES6 (use 'esversion: 6') or Mozilla JS extensions (use moz)."
2  const ctx = canvas.getContext("2d");
   Column: 1 "'const' is available in ES6 (use 'esversion: 6') or Mozilla JS extensions (use moz)."
3
4  console.log(canvas);
5
6  const game = new Game(canvas, ctx);
   Column: 1 "'const' is available in ES6 (use 'esversion: 6') or Mozilla JS extensions (use moz)."
7
8  const controls = new Controls(game);
   Column: 1 "'const' is available in ES6 (use 'esversion: 6') or Mozilla JS extensions (use moz)."
9
10 document.addEventListener("keydown", controls.keyDownHandler.bind(controls), false);
11 document.addEventListener("keyup", controls.keyUpHandler.bind(controls), false);
12
13 game.startGame();
```

- *'components/Controls.js':*

components/Controls.js

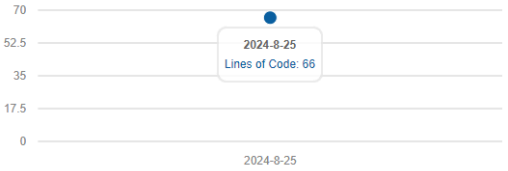
Maintainability

71.90



Lines of code

66



Difficulty

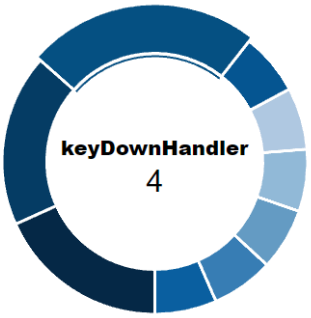
22.62

Estimated Errors

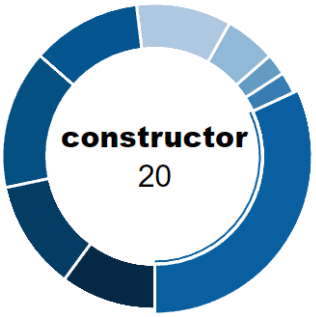
0.73

Function weight

By Complexity



By SLOC

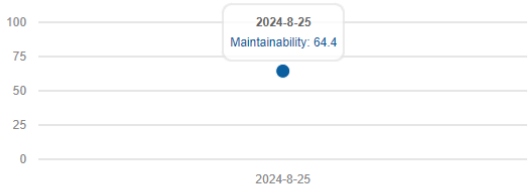


- *'components/Draw.js':*

components/Draw.js

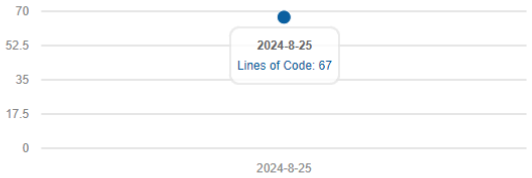
Maintainability

64.40



Lines of code

67



Difficulty

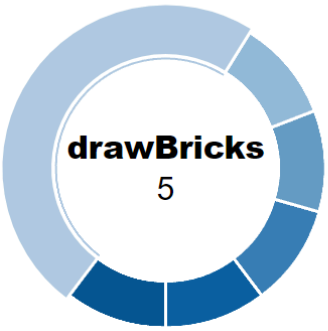
30.13

Estimated Errors

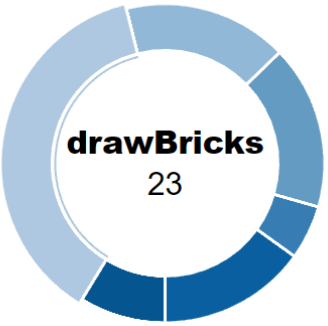
0.82

Function weight

By Complexity



By SLOC

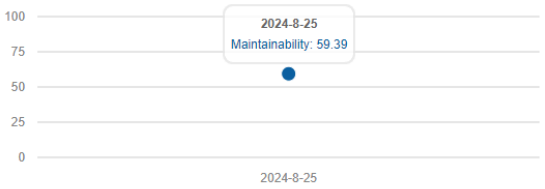


- *'components/Game.js':*

components/Game.js

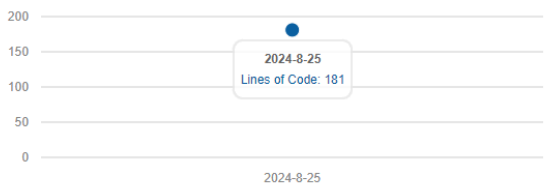
Maintainability

59.39



Lines of code

181



Difficulty

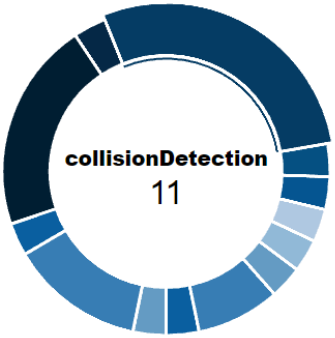
91.14

Estimated Errors

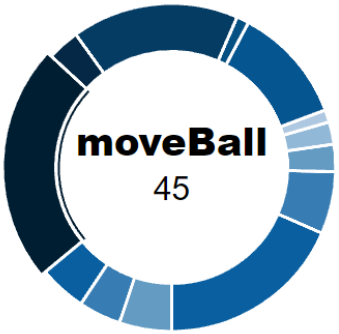
2.57

Function weight

By Complexity



By SLOC



5) Conclusões:
    - O arquivo *game.js* possui o maior número de linhas de código, maior dificuldade e maior número de prováveis bugs, mostrando que será o arquivo com manutenção mais difícil, tanto para entendê-lo quanto modificá-lo.
    - O arquivo *main.js* apresenta uma manutenibilidade alta e com poucos bugs prováveis, mostrando que dificilmente precisará de alguma alteração para correção de erros.
    - Os arquivos *draw.js* e *controls.js* demonstram dificuldade baixa, logo será mais fácil a adição de novas features relacionadas a estes dois arquivos ao programa.
    - O método *collisionDetection* no arquivo *game.js* possui complexidade alta, podendo dificultar seu entendimento devido ao alto número de possíveis rotas.
    - O método *moveBall* no arquivo *game.js* possui o maior número de linhas dentre todos os métodos do programa, também podendo dificultar seu entendimento e manutenibilidade.
    - O método *drawBricks* do arquivo *draw.js* possui baixa complexidade e poucas linhas, facilitando sua alteração.

OBS: os relatórios completos se encontram no git.