Drunk Walking

Relatório

Laura Kácser - 11200643

Matteus Legat - 14100861

Introdução

A aplicação desenvolvida utiliza a técnica de Simulação de Monte Carlo para realizar o cálculo de um caminho aleatório percorrido por um corpo, no caso, um bêbado que realiza passos em ângulos aleatórios.

Ferramentas utilizadas

A implementação da aplicação foi feita utilizando a linguagem de programação JavaScript, que roda nos navegadores modernos de hoje, garantindo uma grande compatibilidade com diversos sistemas, bem com uma interface amigável ao usuário final.

Para a plotagem dos dados em gráficos, foi utilizada a biblioteca Charts.js, que é simples e flexível, além de ser Open Source. A biblioteca foi importada à aplicação no arquivo *lib/js/chart.bundle.min.js*.

O design da interface gráfica foi feito com a utilização de CSS, utilizando o framework open source Bootstrap, que oferece uma grande quantidade de componentes com aparência agradável.

Desenvolvimento

Para a representação de um estado da simulação, foi implementada a classe *State* (*js/state.js*). A classe possui três variáveis, uma que indica qual é o passo (*step*), e outras duas que representam as coordenadas do bêbado (*x* e *y*).

A classe *State* possui a função *generateNextState()*, que gera um ângulo aleatório e calcula novas coordenadas, retornando um novo estado do sistema, após um novo passo.

Ela também possui outras funções para calcular a distância percorrida, a distância estimada e a diferença entre elas, assim como uma função que retorna apenas suas coordenadas.

Para a representação de uma simulação, foi implementada a classe *Simulation* (*js/simulation.js*). A classe possui um array de estados, que é inicializado em seu próprio construtor. Esta classe possui apenas métodos para obter dados de seu último estado, assim como dados prontos para a plotagem no gráfico.

Para a representação de um histograma, foi implementada a classe *Histogram* (*js/histogram.js*). Seu construtor recebe um array com mais de uma simulação e as separa em classes de acordo com a diferença entre as distâncias do estado final. Esta classe possui apenas métodos que retornam dados prontos para a plotagem no gráfico.

Para o controle da interface, foi implementada a classe *UserInterface* (*js/userInterface.js*). Ela possui somente métodos para controlar a interface, que mostram/escondem os gráficos, atualizam seus dados, validam e lêem os parâmetros.

O método principal que é executado quando a simulação é iniciada está implementado no arquivo *js/main.js*. Ele obtém os parâmetros da interface e instancia o número de passos e replicações passados por parâmetro. É ele quem decide se deve ser feito um histograma, ou os outros dois gráficos, e então chama métodos da interface para atualizá-la.

Para a representação de cada um dos tipos de gráficos, foram criadas três classes na pasta *js/charts/*. Elas possuem a configuração específica de cada tipo de gráfico e métodos para atualizar seus dados, que são repassados para a biblioteca *Charts.js*.