Trabalho - Implementação de um jogo em Racket

Crie um jogo lúdico para o ensino de programação da linguagem racket de modo que ele seja educativo, envolvente e eficaz na transmissão de conceitos de programação. Elementos-chave em um jogo com esse propósito:

1. Desafios Graduados:
    a. Comece com desafios simples e aumente gradualmente a complexidade à medida que os jogadores progridem.
    b. Cada desafio deve abordar um conceito específico de programação.
    c. Devem ser abordados pelo menos 4 conceitos específicos de linguagem de programação Racket.
2. Interatividade:
    a. Permita que os jogadores interajam diretamente com o ambiente de programação.
    b. Ofereça uma interface de programação amigável e intuitiva.
3. *Feedback* Imediato:
    a. Forneça *feedback* imediato sobre o código inserido.
    b. Destaque erros e forneça sugestões de correção.
4. História ou Narrativa:
    a. Integre uma história ou narrativa que motive os jogadores a progredirem.
    b. Personagens e enredos podem tornar a aprendizagem mais atraente.
5. Recompensas e Conquistas:
    a. Inclua recompensas ou conquistas para reconhecer o progresso dos jogadores.
    b. Isso estimula a motivação para o aprendizado de racket.
6. Ferramentas de Ajuda:
    a. Forneça ferramentas de ajuda, como tutoriais, dicas contextuais, documentação integrada.

É importante ressaltar que o trabalho é individual ou em dupla, e que os resultados alcançados no trabalho deverão ser apresentados em slides como parte do processo de avaliação da disciplina.

CRITÉRIOS DE AVALIAÇÃO:

Clareza lógica: a lógica nos programas deve ser coerente e fazer sentido.

Utilização das receitas de projeto: as funções devem ser escritas utilizando as receitas de projeto.

Testes: Criar testes unitários para todas as funções.

Corretude e completude: criar um teste principal que execute os testes unitários, contemplando os requisitos definidos.

Boas práticas de programação: o código deve estar bem escrito e organizado; os recursos da linguagem devem ser usados corretamente.

Complexidade: irá avaliar o grau de complexidade da implementação e sua respectiva codificação (número de linhas de código, quantidade de funções, etc).