Universidade de Brasília - UnB
Data: 02/12/2018
Métodos de Programação - 2/2018
Alunos: Amanda Aline Figueiredo Carvalho - 15/0115997
Bruno Justino Garcia Praciano - 13/0006912
Jéssica da Silva Oliveira - 13/0028983

PROJETO FINAL DE MÉTODOS DE PROGRAMAÇÃO
LAUDO FINAL

O objetivo do trabalho era a criação de um jogo de estratégia, que foi nomeado como Duelo das Tropas. As checklists foram baseadas nas utilizadas nos trabalhos anteriores, mas com foco para o jogo específico.

CHECKLIST - DESIGN DO SOFTWARE

Identificar os requisitos funcionais e não-funcionais: deve ser feito levantamento de requisitos para que seja adaptado o design ao escopo do projeto.

- Foi feito levantamento dos requisitos de forma correta e que abrangesse todo o escopo do projeto

Elaborar design de interface de usuário: documentar os passos das telas que o usuário atuará.

- Foi feita documentação com o passo-a-passo do programa desenvolvido. Consta no ReadME do Github.

Flexibilizar o layout para adaptações: o layout deve ser feito de forma a abranger as adaptações que surgirem no decorrer do desenvolvimento do projeto.

- Foi feito um escopo que fosse aberto a possíveis alterações no decorrer do projeto

Cores: as cores presentes nas telas que o usuário terá acesso, devem condizer com os anseios, mas ao mesmo tempo, prezar pela usabilidade e ser acessível para todos os públicos.

- Foram escolhidas cores padrões da biblioteca gráfica utilizada (ncurses), para que o jogo, embora simples, pudesse ter cores vivas para o jogador

Excesso de texto: o excesso de texto pode prejudicar substancialmente o layout do software, pois pode não ser tão amigável no produto final.

- Foi dada atenção para que a interface, embora precisasse conter as informações essenciais do jogo, não ficasse muito sobrecarregada.

CHECKLIST - CODIFICAÇÃO

1. **Comentar enquanto redige o código:** o código deve ser comentado durante a escrita do programa, pois evita que o programador perca a linha de raciocínio, durante a parada ou que o código vire legado por falta de documentação.
   - Os comentários foram feitos enquanto se fazia cada função, para facilitar a documentação

2. **Fazer o código por etapas:** a programação deve ser feita por meio de testes, trabalhando com etapas de código e não tudo de uma vez, pois pode gerar problemas.
Ex: o programador deve programar módulo por módulo, testando suas partes, para que não crie a situação de não consertar o erro.
   - O código foi feito por etapas, levando um alto número de horas para conclusão.

3. **Não recomenda muitas linhas de código de uma vez:** devido ao fato do problema de criar mais código baseado em um trecho que está errado anteriormente, recomenda-se poucas linhas e muito teste.
   - As linhas de código foram desenvolvidas módulo a módulo para controlar os testes.

4. **Fazer os asserts durante a programação:** os asserts devem ser feitos enquanto os trechos são escritos, parte por parte.
   - O código está funcional e os testes feitos à parte.

5. **Manter indentação do código bem feita:** a indentação do código bem feita é uma das boas práticas de programação que permite que outro programador dê prosseguimento ao seu código, sem que os módulos estejam visualmente desconectados um com o outro.
   - A indentação do código está bastante legível e de acordo com os padrões.

6. **Mitigar problemas fazendo desenvolvimento orientado a testes:** o desenvolvimento orientado a testes evita problemas sem resolução e bugs não esperados, pois a cada trecho de código, são realizados diversos testes que permitem visualizar como o programa está se comportando.
   - utilizar uma plataforma de testes. No caso foi utilizado o CUnit para desenvolver os testes, devido ao problemas gerados pela execução do Gtest e Catch, junto com o código. Estava ocorrendo um erro referente ao g++, onde assim, como o código é em C, optou-se por usar uma ferramenta de mesma linguagem, mesmo que não esteja nas especificações, pois este erro não pode ser tratado.

7. **Garantia de condição de entrada:** é uma condição que deve ser sempre verdadeira antes da execução de alguma seção de código ou antes de uma operação em uma especificação formal.
   - As condições foram satisfeitas.

8. **Garantia de condição de saída:** é uma condição que deve sempre ser verdadeira logo após a execução de alguma seção de código ou após uma operação em uma especificação formal.
   - As condições foram satisfeitas.