
# UFSCar Summer Tour

Antonio Carlos Falcão Petri 586692
José Antônio dos Santos Júnior 586765
José Vitor de Carvalho Aquino 609170
Tiago Bonadio Badoco 586722

São Carlos
Junho/2015

## Autores:

Antonio Carlos Falcão Petri (falcaopetri@gmail.com) – Implementação do Jogo

José Antônio dos Santos Júnior (jusantosjr@hotmail.com) – Desenvolvimento Gráfico

José Vitor Aquino (jvcaquino95@gmail.com) – Implementação das Estruturas de Dados

Tiago Bonadio Badoco (tiago.badoco@gmail.com) - Documentação

## Funcionamento do Jogo:

O objetivo do jogo é chegar até determinado ponto do campus da UFSCar de São Carlos. Para isso o jogador parte do Restaurante Universitário e sempre tem dois caminhos para seguir o da direita ou o da esquerda. Cada escolha leva o jogador a um ponto diferente que pode ser um destino final ou um ponto que lhe proporcione duas novas escolhas.

É importante destacar que o jogador não pode voltar no caminho, ou seja, só pode avançar. Cada caminho para um ponto destino é único, ou seja, o usuário deve fazer um caminho especifico para vencer. O local de destino é sorteado de maneira aleatória, toda vez que uma partida é iniciada.


Imagem 1 - Menu principal

## Controles:

Mouse – O mouse controla todas as interações com os menus do jogo, selecionando as opções desejadas.

Seta Esquerda – Seleciona o caminho da esquerda.

Seta Direita – Seleciona o caminho da direita.


Imagem 2 - Gameplay

## Ferramentas:

- Codeblocks;
- Allegro 5.1;
- Photoshop CS5;
- Astah Professional;
- Fruity Loops Producer Edition 11;
- STL, C++ Standard Template Library.

## TAD's

A implementação do jogo utilizada utiliza-se de 2 estruturas de dados para organizar os principais elementos do jogo, uma Árvore Binária e um Map.

## Como a Árvore é utilizada e implementada?

A Árvore Binária é definida por um Nó (TreeNode) que guarda um valor e a uma informação que identifica suas subárvores. Para operar sobre a Árvore, criou-se uma *friend class* BTUtil, com as manipulações necessárias para o jogo.

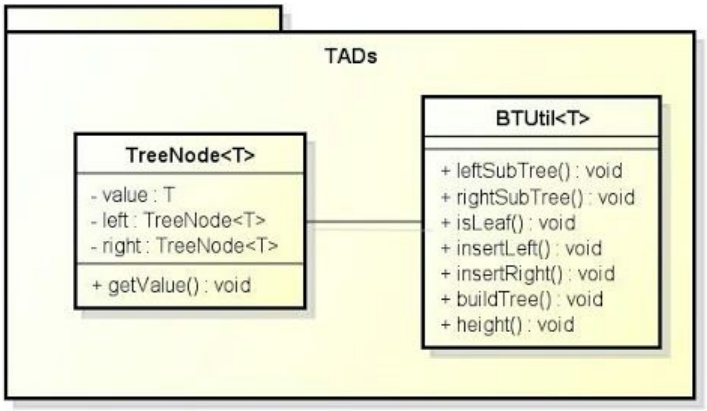


Imagem 3 - Diagrama de Classes: Árvore Binária

## Como o Mapa é utilizado?

Um Mapa é uma estrutura de dados que permite o armazenamento de elementos "indexados" por uma chave, normalmente única. Externamente, e para fins didáticos, se parece com um vetor que, ao invés de ter índices numéricos e acessar o elemento "na posição I", permite ter qualquer tipo de objeto como índice e acessa o elemento "com a chave K".

Internamente, entretanto, os mapas possuem comportamentos diferentes. Eles costumam ordenar seus dados em relação à chave e, para manter eficiência nas operações de inserção, consulta e remoção, são implementados com árvores binárias balanceadas.

Na implementação desse jogo, foi utilizado o std::map, um container da STL. Dessa forma, pode-se definir identificadores às imagens utilizadas e manipuladas pelo jogo. São utilizados dois maps. Um relaciona o identificador da imagem ao endereço (relative-path) de seu arquivo no sistema. O outro, relaciona um identificador (o mesmo, por comodidade e coerência) à um objeto da Classe IDrawing.

Dessa forma, reúne-se todas as imagens do sistema em uma única estrutura de dados, que assegura velocidade na manipulação desses objetos. Ex:

```
systemImages["name"].getBitmap();
```

## Arquitetura do Software:

A disposição geral do jogo baseia-se de criação de uma TourGUI, que encapsula tanto as regras do jogo (Game) quanto a interpretação gráfica dada a ele. Para tal, criou-se o conceito de Telas (Screens), que possuem um sub-conjunto de imagens desenháveis.

Dessa forma, não é necessário identificar qual Tela atual está sendo exibida, pois todas possuem a mesma interface (de métodos). A Tela do Jogo (PlayScreen), por possuir uma lógica de funcionamento diferenciada, é uma especialização de uma Screen.

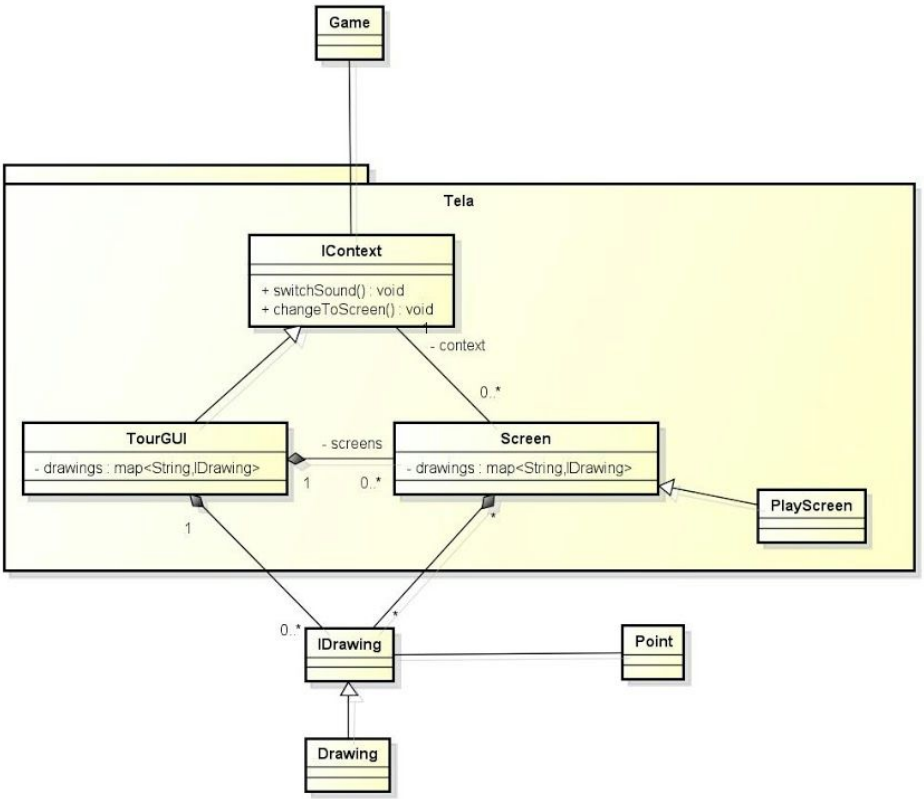


Imagem 4 - Diagrama de Classes: Visão simplificada do Jogo

## Observações e Comentários:

Essa seção reúne pequenas informações quanto ao processo de desenvolvimento do jogo.

- Para atribuir um ícone ao executável, seguiu-se o procedimento descrito em https://www.allegro.cc/forums/thread/601721Monolith;
- Utilizou-se a função setResourceArchive() no arquivo TheLastTooFast.cpp. Tal função está descrita em https://www.allegro.cc/forums/thread/614268, e faz com que o working-directory da

aplicação seja mudado para a pasta em que se encontra o executável. Assim, pode-se utilizar relative-paths para referenciar os arquivos usados pelo jogo;

- A Arquitetura de Screens proposta não foi suficiente para a representação abstrata de uma Tela, tendo-se que criar artifícios internos para a correta manipulação dos elementos. De toda forma, ela pode ser revista e refinada, possivelmente, enquadrando-a em um Design Pattern.
- De fato, tal abordagem permitiu um maior reúso do código central do jogo: o Trabalho 2, Tra1n, que também utiliza-a, foi facilmente adaptado para gerar esse outro jogo;
- Foi utilizado o Allegro 5.1, a primeira versão da biblioteca a ter suporte à reprodução de vídeos. Para tal, foi necessário muita pesquisa e testes experimentais, uma vez que essa versão do Allegro ainda está em desenvolvimento. Esse fato, aliás, pode acarretar em pequenos problemas durante a reprodução dos vídeos.