Projeto 1 - Interactive Fiction

UTFPR/DACOM 2017

BCC35A-Linguagens de Programação

Projeto 1: Interactive fiction

- Jogo de narrativa com múltiplos caminhos
  - Ênfase em:
    - narrativa
    - possibilidades de influenciar a história
    - múltiplos caminhos (alguns podem intersectar)
    - você "navega" pelo mapa pelo uso de comandos
- Inspiração:
  - http://textadventures.co.uk/
  - https://en.wikipedia.org/wiki/Interactive_fiction

Formato e Requisitos

- História interativa
  - Apenas escolhe opção
  - Menu em texto simples
- Narrativa
  - Pode usar livro de preferência como base
  - Múltiplas possibilidades de caminhos na história
- Quesões de implementação
  - Qual estrutura de dados utilizar?
    - ★ Hashmap para organizar salas e comandos
    - ★ O comandos podem levar a outras salas

Linguagens

- OBS: será necessário **instalar compilador/interpretador** no computador
  - para ler arquivos das salas
  - para tocar sons via terminal
  - para gerar arquivo dot de visualização do grafo (mapa) das salas

1. Dart - Emerson Yudi
2. JavaScript - Leonarco
3. Go
4. Julia
5. C# - Kelvin
6. Kotlin - Marlon
7. Lua - Matheus
8. Ruby - Guerra
9. Swift - Vitor

Avaliação

- **Cópias**: Qualquer tipo de cópia (trabalhos de colegas, internet, etc) anulará o trabalho
  - Seja por porções de código ou pelo trabalho completo
- **Entrega**: código fonte pelo Moodle
- **Data**: a definir
- **Apresentação**: em aula
- **Ferramentas**: editores e compiladores/interpretadores de cada linguagem (**instalar em seu PC**)
- **Visualização**: gerar código em texto puro para visualizar grafo do "mundo" do jogo em Graphviz
- **Critérios**:
  - Mínimo: implementar o jogo -> nota mínima
  - Diferencial: qualidade e recursos adicionais -> incrementa até a nota máxima

Requisitos

- Mínimo
  - 20 salas
  - Número variado de comandos por salas
  - Jogador pode perder a partida
  - Execução de áudio (som ambiente para as salas ou efeitos sonoros) via comandos do terminal
  - Salas devem ser configuradas em arquivo(s) texto (puro, JSON ou XML)
  - Gerar arquivo dot para visualização com graphviz
- Diferencial
  - Esmero com interface e mensagens do jogo
  - Complexidade e qualidade da história
  - **NOVAS FUNCIONALIDADES**

Visualização do mundo do jogo

- Gerar código na sintaxe **dot** de forma que possamos visualizar o grafo formado pelas salas
- Template da estrutura de diagrama a ser montado para Graphiz
  - http://www.graphviz.org/content/datastruct
- Visualizador online de código graphviz (dot)
  - http://www.webgraphviz.com/

Exemplo de implementação: estruturas

```
// estrutura da sala
class Sala {
    constructor(description, options) {
        // texto da sala
        this.descrition = description
        // array de comandos possiveis (chaves do hash)
        this.options = options
    }
}

// estrutura do comando
class Comando {
    constructor(comando, resultado, chave='nenhuma') {
        // texto do comando
        this.texto = comando
        // resultado caso comando seja selecionado
        this.resultado = resultado
        // chave da proxima sala ou nenhuma se caminho impossível
        this.chave = chave
    }
}
```

Exemplo de implementação: o mundo

```
// sala
entrada = new Sala(
    'Você abre os olhos e vê uma porta aberta...',
    [
        new Comando('Entrar na porta', 'Você passou pela porta', 'primeira_sala'),
        new Comando('Abrir o portão dos fundos', 'A maçaneta está quebrada'),
        new Comando('Seguir para o jardim',
                    'Você encontra um jardim morto', 'jardim')
    ]
)

// sala
primeira_sala = new Sala(
    'Ao abrir a porta você se depara com uma estátua...',
    [
        new Comando('Voltar à primeira sala',
                    'A porta fechou-se e você não consegue voltar'),
        new Comando('Empurrar estátua', 'É muito pesada para sair do lugar'),
        new Comando('Pressionar botão',
                    'Uma porta se abriu na parede e você seguiu por ela',
                    'primeira_sala')
    ]
)
```

Exemplo de implementação: testando

- Construindo o "mundo" do jogo

```
// usando array associativo
salas = []
salas['entrada'] = entrada
salas['primeira_sala'] = primeira_sala

console.log(salas['primeira_sala'])

// usando HashMap
map = new Map()
map.set('entrada', entrada)
map.set('primeira_sala', primeira_sala)

console.log(map.get('entrada'))
```