Projeto 1 - Text Adventure

UTFPR/DACOM 2018

BCC35A-Linguagens de Programação

Text Adventure

- Jogo do gênero Adventure (puro) em modo texto
  - Populares nos anos 80
  - Ênfase em narrativa, exploração e puzzles
- Mais sobre text-based games
  - https://en.wikipedia.org/wiki/Text-based_game
  - https://en.wikipedia.org/wiki/Interactive_fiction
- Jogos do gênero:
  - http://textadventures.co.uk/

Text Adventure: características gerais

- Jogo em modo **texto** com foco em narrativa
  - Interação via linha de comandos: use, pick, check...
  - Mundo composto por "cenas"
    - ★ Interligadas
    - ★ **Não é permitido** utilizar **biblioteca** ou **hash**: implementação da TAD faz parte do trabalho
  - Jogo deve ter final
    - ★ Múltiplos caminhos para se chegar ao final
  - Forte ênfase em narrativa: seja criativo
    - ★ Possibilidades de influenciar a história
    - ★ Múltiplos caminhos (alguns podem intersectar)
  - Gerenciamento simples de inventário
    - ★ Obter objetos da cena
    - ★ Usar objetos do inventário em objetos da cena

Gameplay

- O jogo apresenta o título a descrição da cena para o jogador
- O mundo do jogo é organizado em cenas. O jogador navega entre as cenas ao executar o comando **use** em um OBJECT
- Os objetos da cena com os quais o jogadaor pode interagir são sempre descritos em MAIÚSCULAS
- A única forma de passar de uma cena para outra é atraveś da interação com objetos da cena (SCENE_OBJECT)
  - Pelo uso de objeto da cena
    - /> **use SCENE_OBJECT**
    - Ex: use PORTA.
    - Resultado: Você abriu a Porta e passou para o corredor.
  - Pelo uso de um objeto do inventário em um objeto da cena
    - /> **use INVENTORY_OBJECT with SCENE_OBJECT**
    - Ex: use PEDRA with JANELA
    - Resultado: A pedra quebrou a janela e você passou por ela.
- Objetos são usados para resolver quebra-cabeças (puzzles)

Gameplay: exemplo

```
Cena 1: O Início
Você abre os olhos e está em quarto pequeno, com pouca iluminação.
Você vê um INTERRUPTOR na parede que está a sua frente.
Ao lado oposto, há uma JANELA. Abaixo de sua cadeira há uma PEDRA.

/> use INTERRUPTOR
O interruptor esta quebrado e sem efeito.

/> use JANELA
Não há efeito.

/> check JANELA
Um dos vidros está rachado.

/> get PEDRA
A pedra está no inventário.

/> inventory
PEDRA, MANUAL

/> use PEDRA with JANELA
A pedra quebrou a janela e você passou por ela.
(deve mudar de cena, mostrando o novo texto)
```

Comandos

- Texto da cena deve enfatizar objetos interagiveis em MAIÚSCULO
- Implementar parser de comandos básicos
- Comandos do jogo
  - inventory
  - use SCENE_OBJECT
  - use INVENTORY_OBJECT with SCENE_OBJECT
  - check OBJCT (qualquer tipo)
  - get SCENE_OBJECT
- Comandos do sistema
  - help
  - save NOME_SAVE
  - load NOME_LOAD
  - newgame

Comandos do Jogo

- Inventário de itens
  - /> **inventory**
- Interagir com algo (abrir, entrar, acionar)
  - /> **use SCENE_OBJECT**
  - Como o click do mouse em um Point & Click
  - ex: use PORTA -> mostra resposta e vai para outra cena
- Combinar objetos do inventário
  - /> **use INVENTORY_OBJECT with SCENE_OBJECT**
  - pode usar objeto do inventário em objeto da cena
    - ★ resolve puzzle
  - deve checar se inventário possui o INVENTORY_OBJECT
- Descrever objeto (tanto da cena quando do inventario)
  - /> **check OBJECT**
- Obter OBJECT para o inventário (que é coletável)
  - /> **get INVENTORY_OBJECT**

Comandos do sistema

- Instruções e Ajuda de comandos
  - /> **help**
  - Fornece instruções gerais do jogo
  - Fornece lista e descrições dos comandos
- Salva estado atual do jogo
  - /> **save NOME_SAVE**
  - Salva em arquivo: cena corrente, objetos do inventário e estado de todos os objetos das cenas
  - NOME_SAVE indica nome do arquivo
- Carregar estado do jogo
  - /> **load NOME_SAVE**
  - Carrega de arquivo os dados salvos
  - NOME_SAVE indica nome do arquivo
- Reiniciar jogo
  - /> **newgame**
  - Pede confirmação ao usuário (S/N)

Requisitos e Critérios

Linguagens

- Será necessário **instalar compilador/interpretador** no computador
  - UI em **modo texto**
  - leitura das cenas e objetos
  - escrita e leitura do estado do jogo (save/load)
  - para tocar sons via terminal
- Linguagens Possíveis

Object Pascal	Visual Basic (Mono)
Go	C# (Mono)
Rust	TypeScript
Lua	Ruby
Perl	Coffee Script
Swift	Action Script
Kotlin	D
Groovy	Dart

Requisitos

- Mínimo
  - 10 salas
  - Número variado de objetos por salas
  - Jogador pode "perder" e reiniciá-la
  - Salas devem ser configuradas em arquivo(s) texto
    - ★ Puro, JSON ou XML
- Diferencial
  - Esmero com interface e mensagens do jogo
  - Complexidade e qualidade da história
  - **NOVAS FUNCIONALIDADES**
    - ★ Sejam criativos

Avaliação

- **Acompanhamento**: a toda aula professor irá registrar status do desenvolvimento do trabalho. A aula **deve** ser usada para produzir o trabalho. Desvios de foco ou inconveniências (ex: falar alto, atrapalhando os demais) acarretarão em penalidades (na nota final).
- **Cópias**: Qualquer indício de cópia anulará o trabalho
  - Seja por trabalhos atuais/anteriores de colegas, internet, etc...
  - Seja por porções de código ou pelo trabalho completo
- **Entrega**: código fonte pelo Moodle
- **Data**: ver plano de ensino
- **Apresentação**: em aula
- **Ferramentas**: editores e compiladores/interpretadores de cada linguagem (**instalar em seu PC**)
- **Critérios**:
  - Mínimo: implementar o jogo -> nota ok
  - Diferencial: qualidade e recursos adicionais -> incrementa até a nota máxima

Projeto e Implementação

Estrutura do jogo

- O mundo do jogo é composto de um vetor de Cenas
- Cada cena possui um vetor de objetos, que podem ser SCENE_OBJECT ou INVENTORY_OBJECT
- Quando um objeto é coletado (comando **get**), uma cópia é criada e adicionada ao vetor dentro do **registro Inventario**

```
registro Jogo
    cenas: Vetor de Cenas
    cena_atual: Inteiro // index da cena no vetor

registro Cena
    id: Inteiro // index da cena no vetor
    titulo: Texto
    descricao: Texto
    itens: Vetor de Objetos

registro Inventario
    itens: Vetor de Objetos
```

Estrutura do jogo

```
registro Objeto
    // para diferenciá-los (SAVE_GAME)
    id: Inteiro
    // SCENE_OBJECT = 0, objeto de interação da cena. Pode ser aplicado um
    // INVENTORY_OBJECT sobre ele para solucionar puzzle
    // INVENTORY_OBJECT = 1, pode ser obtido pelo comando "get" e
    // vai para o inventário ("obtido" para a ser true)
    tipo: Inteiro
    nome: Texto
    descricao: Texto
    resultado_positivo: Texto // exibido após uso do comando correto
    resultado_negativo: Texto // indica o uso de comando incorreto
    // guarda o comando correto que o "soluciona".
    // Ex: use INTERRUPTOR; get PEDRA; use PEDRA with JANELA
    comando_correto: Texto
    // index da cena para qual este objeto leva caso o comando_correto for
    // usado ou -1, caso não leve a lugar algum
    cena_alvo: Inteiro // para SCENE_OBJECT
    // indica se já houve comando anterior que o resolveu
    // (para navegar em salas já visitadas/solucionadas)
    // Ex: uma porta que precisada de chave e isso já foi resolvido
    // Logo, se voltar a interagir, basta usar "use SCENE_OBJECT"
    resolvido: Booleano // para SCENE_OBJECT (SAVE_GAME)
    // indica se objeto já foi obtida da cena, isto é, está no inventário
    obtido: Booleano // (SAVE_GAME)
```

Estrutura do jogo: considerações

- Se usuário executa comando **get OBJECT** e obtido == -1 então o objeto já foi coletado e está no inventário (ou já foi usado)
- Após um objeto do inventário ser utilizado, ele permanece como obtido no vetor da cena e é removido do vetor do inventário
- Essencialmente há dois tipos de objetos: INVENTORY e SCENE
- **INVENTORY_OBJECT**: vai para o inventário com **get** e usualmente é usado em objeto da cena para solucionar puzzles (comando **use INVENTORY_OBJECT with SCENE_OBJECT**)
- **SCENE_OBJECT**: objeto que permite interação funcional (faz algo significativo na cena) ou apenas decorativa
- Quando um comando é digitado, é preciso checar se o(s) objeto(s) envolvido(s) está(ão) na **cena** (se SCENE_OBJECT) ou no **inventário** (INVENTORY_OBJECT)

```
registro Objeto
    id: Inteiro // (SAVE_GAME)
    tipo: Inteiro
    nome: Texto
    descricao: Texto
    resultado_positivo: Texto
    resultado_negativo: Texto
    comando_correto: Texto
    cena_alvo: Inteiro // para SCENE_OBJECT
    resolvido: Booleano // para SCENE_OBJECT (SAVE_GAME)
    obtido: Booleano // para INVENTORY_OBJECT (SAVE_GAME)
```

Implementação do Mundo do jogo

- O registro Objeto é fundamental para o funcionamento do jogo e a navegação entre cenas
- Backtracking: pode ser necessário voltar a cenas anteriores para progredir
  - Não é obrigatório colocar no jogo
  - Pode usar múltiplas cenas relacionadas
    - ★ Ex: precisa obter o objeto de uma cena para usar em outra
- Cenas como consequencias de anteriores
  - Você pode criar cenas semelhantes em NODES diferentes do grafo.
  - Tais cenas podem passar a impressão de que o jogador volta a cena original, mas que está alterada
  - Ex: usar ISQUEIRO e atear fogo no quarto. A nova cena descreve o mesmo quarto alterado, em chamas