Centro Acadêmico de Ciência da Computação
DIN - Departamento de Informática

Wellington Tatsunori Asahide

Universidade Estadual de Maringá (UEM)

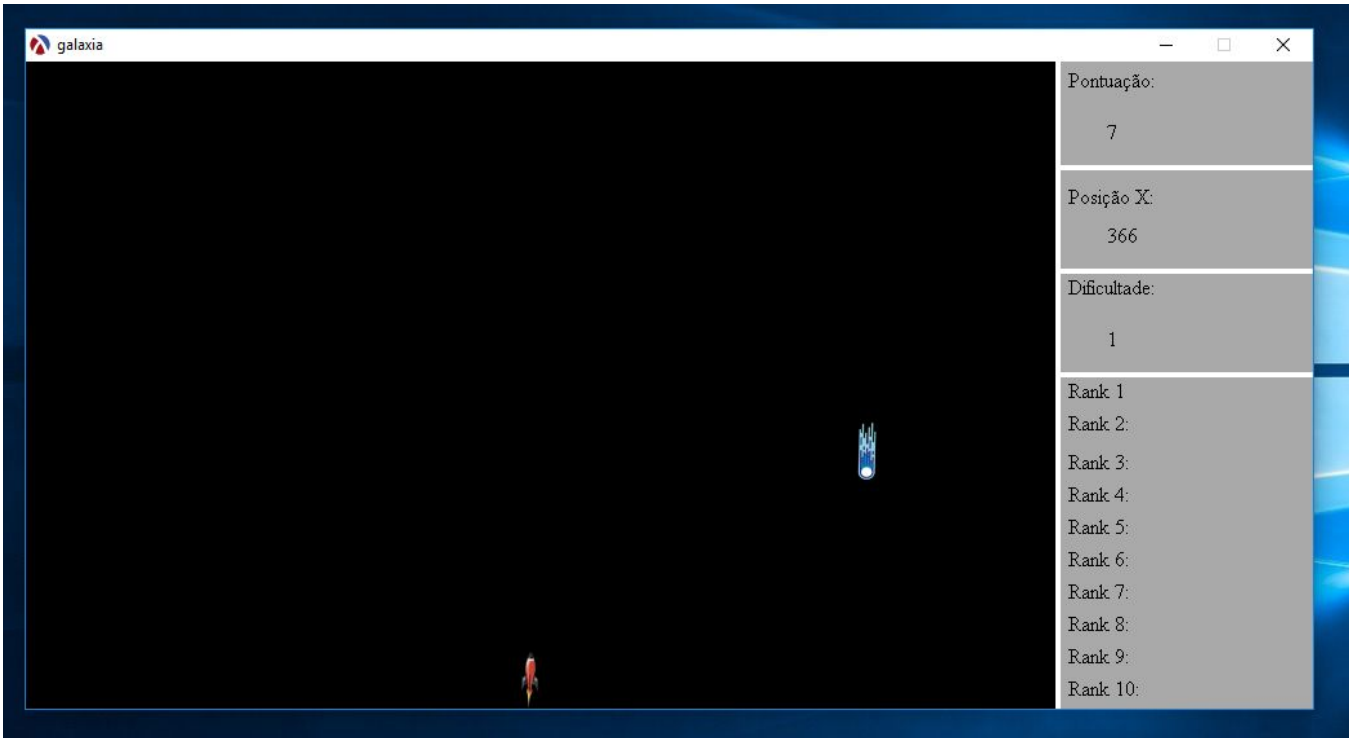
Trabalho de paradigma de programação lógica e funcional

Sumário

Space destiny

O jogo implementado tem como objetivo desviar dos meteoros que vão chegando ao seu encontro através dos comandos do mouse. Conforme a quantidade de meteoros que você desviou, eles começam a se tornar mais rápidos, dificultando assim o jogo.

Segue abaixo uma imagem para entender como funciona:



E as imagens do foguete e o cometa:

Solução do jogo

O jogo não possui fim, portanto não possível desenvolver um algoritmo que o resolva, porém foram pensados em 4 IA's capazes de jogar o space destiny, elas serão apresentadas mais abaixo no próximo tópico

Considerações requisitadas

Variáveis do jogo

(define nome "NAname")

Esta variável é utilizada para guardar o nome do jogador, no caso, a pessoa digita o seu nome em um interface com um text field, e a string nesse text é salvo na variável nome. Caso a pessoa não digite um nome, por padrão, ele será "NAname".

Existe uma variável x, que serve para guardar valores do plano cartesiano x do jogo, segue abaixo as formas em que ela foi utilizada:

- Simular a movimentação do cometa
- Simular a movimentação da nave
- Evaluar o choque da nave com o cometa
- Definir a posição em que o cometa irá cair
- Limitador a esquerda/direita da movimentação da nave

A variável y foi utilizada em grande parte em conjunta com a x, portanto, muitas coisas são similares:

- Simular a movimentação do cometa
- Simular a caída do cometa
- Evaluar o choque da nave com o cometa

posx foi utilizado para saber a posição em que a nave se encontra, logo, também foi utilizada para evaluar o choque da nave com o cometa, assim como a variável x

t e esp foram utilizadas para manipular a velocidade em que os meteoros caem

cont cont2 e cont3 servem para marcar a pontuação, e level

Definição do nome do jogador

Para definir o nome do jogador, foi utilizado a seguinte interface (segue abaixo o código):

```
(define inicio (instantiate dialog% ("Bem vindo capitão")))

(define jogador (new text-field% [parent inicio] [label "Nome:"]))

(define panel (new horizontal-panel% [parent inicio]
                                     [alignment '(center center)]))
(new button% [parent panel] [label "Proximo"]
   [callback (lambda (button event)  (send inicio show #f) (salva-nome (send
jogador get-value)) (sleep 1))])
(send inicio show #t)
```

Em que ao clicar o botão em próximo, ele chama a função salva-nome, definindo um novo valor para a variável nome. Segue abaixo o código da função salva-nome:

```
(define (salva-nome str)
  (set! nome str))
```

Jogador versus software

O jogo em si é apenas contra o software, isto é, o algoritmo gera aleatoriamente as posições em que o meteoro irá cair, segue abaixo a explicação do algoritmo:

```
(set! x (random 740))
(coso x y)
```

Primeiro é setado um valor aleatório para x, tal que seja menor que 740 (menor que 740 devido a resolução do jogo). Após isso é chamado a função coso, segue abaixo o código dela:

```
(define (coso x y)
  ((draw-pixmap oculta) "cometa.bmp" (make-posn x (+ y 50)))

  ((draw-pixmap oculta) "escuro.bmp" (make-posn x y));aqui
)
```

Primeiramente é desenhado o cometa, e após isso é desenhado uma imagem preta do mesmo tamanho, assim, isto nos possibilita a simulação do cometa

Para a movimentação da nave, é dependente da entrada do mouse, segue abaixo o código para melhor compreensão:

```
  (if (equal? (query-mouse-posn janela) #t)
      (teclado (posn-x (query-mouse-posn janela)) y x t esp cont cont2 cont3)
      (begin
        (quadro (posn-x (query-mouse-posn janela)) 'L) ;definição da posição da
nave
        (teclado (posn-x (query-mouse-posn janela)) y x t esp cont cont2
cont3);lógica do jogo

        )
    )
```

A função quadro serve para simular a movimentação da nave, assim como a função coso de cometa:

```
(define (quadro x pos)
  (if (equal? pos 'R)
      (begin
```

```
      ((draw-pixmap oculta) "foguete.bmp" (make-posn x 450)))
    (if (equal? pos 'L)
       (begin
         ((draw-pixmap oculta) "foguete.bmp" (make-posn x 450)))
       ((draw-pixmap oculta) "foguete.bmp" (make-posn x 450))
    )
 )
 (copy-viewport oculta janela)
 ((draw-solid-rectangle oculta) (make-posn x 450) 50 50 "black") ;apaga a
nave do quadro anterior, simula o movimento da nave
)
```

A função teclado (recebeu o nome de teclado e não de mouse devido a que primeiramente ele ia ser implementado com comandos do teclado), é encarregada do tratamento da colisão, pontuação, velocidade do meteoro, e salvar a pontuação do jogador.

Ela não será apresentada aqui devido ao seu tamanho a fim de não poluir exageradamente o documento, porém sua complexidade não é elevado, e o código encontra-se bem comentado, portanto acreditamos que você leitor não encontrará dificuldades em entender-lo.

Inteligência artificial

Para isso foram pensadas quatro maneiras, a seguir elas serão descritas:

Primeiro, seria descobrir o valor do eixo x do cometa, assim podemos fazer com que a nave movimente se 50 unidades para a esquerda ou para a direita (o lado depende se irá chegar ao limite da janela). O problema deste algoritmo é que para cada cometa, a nave teria que se deslocar uma vez, apesar da complexidade ser apenas O(1), pois ao saber a posição do meteoro, basta setar a posição da nave como x+/-50.

A segunda opção seria verificar a posição do cometa e da nave, se eles forem colidir, então movimentar a nave x+/-50 para a esquerda ou direita, isso

então evitaria uma grande quantidade de movimentos desnecessários, apesar de ser necessários alguns outros cálculos.

Outra opção seria então definir aleatoriamente uma posição x para a nave, isto poderia ser feito a cada cometa ou a cada certa quantidade de cometa, ou apenas uma vez etc. Mas isso poderia fazer com que o jogo terminasse rapidamente, apesar de fazer uma quantidade de cálculos menores.

A última maneira pensada foi adicionar as posições que o cometa devem cair em uma lista de dez em dez, assim saberíamos a posição em que os próximos dez cometas iriam cair, setando assim de forma segura, um x para a nave. Esta é a maneira que mais consome processos, porém deve ser a que exige menos movimento da nave de forma segura.

Anteriormente foi mencionado em específico o valor 50, isto é devido ao tamanho da nave e do cometa. O algoritmo implementado as IA's estão junto com o código do jogo.

Placar Final

Para o placar final, foi concatenado a variável nome com a variável cont2, segue abaixo o código:

```
((draw-string (open-viewport "Pontuação:" 250 30)) (make-posn 20 15)
(string-append (string-append nome "  ") (number->string cont2)))
                (sleep 3);para o programa por 3 sg
```

Este trecho é executado quando a condição de colisão é atendida

Rank dos 10 melhores

Primeiramente é necessário criar o arquivo, isto foi feito com o código abaixo

```
(define out (open-output-file "file.txt" #:mode 'text #:exists 'append))
```

Caso o arquivo não exista, ele será criado. Após isso é necessário salvar os dados no arquivo, isto foi feito da seguinte maneira:

```
(write (list (string-append (number->string cont2) " " nome )) out)
```

Escreveremos no arquivo, uma lista em que possui as variáveis cont2 e nome concatenadas em formato de string.

Para recuperar os dados do arquivo foi usado o seguinte trecho:

```
(define (get-ranking posicao)
       (let
          ([l1(file->list "file.txt")])
          (define l2 (build-list (length l1) (lambda (x)
             (define aux (string-split (list-ref (list-ref l1 x) 0)) )
              (ranking (string->number (list-ref aux 0)) (list-ref aux 1) )
          )))
         (cond
          [(< posicao (length l1))
          (string-append (number->string (ranking-pontos (list-ref (sort l2 > #:key
ranking-pontos) posicao))) " " (ranking-nome (list-ref (sort l2 > #:key
ranking-pontos) posicao)))]
          [else (string-append " " " ")]
          )
        )
  )
```

Logo, todos os dados de todos os jogadores são salvos, porém é feito a ordenação de acordo com suas pontuações em tempo de execução, e é usado apenas os 10 primeiros.

Talvez seu entendimento seja um pouco complicado, porém o que preciso entender é que, as informações do arquivo são transformadas para a struct do ranking e após isso são transformadas novamente em string para dar o retorno e imprimir. posição serve para selecionar determinado elemento da lista (já ordenada). O método para ordenação foi utilizado o já implementado na biblioteca do racket (-sort lista string>?- ordena por ordem de string, considerando números como strings também). A condicional serve para caso a lista possua menos de 10 elementos, assim ela retornará zero caso não exista.

Para o desenvolvimento deste método foi utilizado a ajuda do Professor Dr. Wagner Igarashi e um compañeiro do curso de Ciência da computação da Universidade Estadual de Maringá.

Por último então, basta fechar o arquivo:

```
(close-output-port out)
```

Referências

http://wigarash.blogspot.com.br/
https://docs.racket-lang.org/
http://es.calameo.com/read/0047422531b892529d528