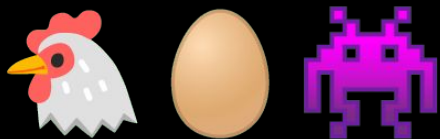


CHICKEN INVADER



Alunos: Diogo Brumassio, RA: 120122 ;
João Vitor Staub Castanho, RA: 117174

Professor: Wagner Igarashi

Explicando o Jogo

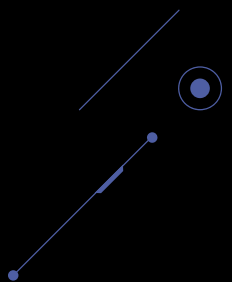
O jogo consiste em um duelo entre uma galinha e uma nave (controlada pelo usuário), a nave pode ser movida para a direita, esquerda, cima e baixo.

Para determinar se o jogador será vitorioso é necessário zerar a vida da galinha (vida vermelha no canto superior) e consequentemente zerar a vida da nave (vida verde localizada na parte inferior da tela) ocasionaria a derrota do jogador.

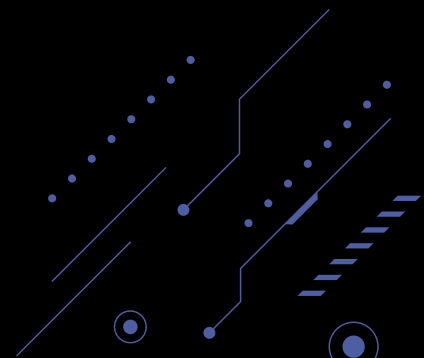
Quando a vida da galinha é zerada o jogo mostra uma mensagem informando a vitória, já quando a vida do jogador é zerada o jogo mostra uma mensagem informando a derrota.



Linguagem



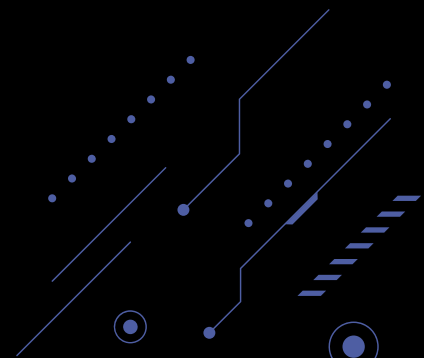
A *Beginning Student* é uma versão pequena do Racket que é adaptada para estudantes iniciantes de ciência da computação.





Arquivos de configuração

São 3 arquivos de configuração, em que cada um para cada dificuldade do jogo (fácil, médio e difícil). Os dados lidos do arquivo são a velocidade da nave, a cor do tiro e o dano que a galinha sofre quando é atingida.



Principais conceitos de linguagens funcionais

O código implementado utiliza vários conceitos de linguagens funcionais, incluindo:

- **Abstração de funções:** as funções são usadas para modularizar o código e torná-lo mais legível.
- **Programação puramente funcional:** o código não tem efeitos colaterais.
- **Uso de expressões matemáticas:** o código utiliza expressões matemáticas para calcular o resultado do jogo.
- **Utilização da notação pré-fixada:** o código utiliza essa notação para realizar as chamadas das funções.

Em resumo, o código implementado no jogo apresenta alguns conceitos típicos de linguagens funcionais.

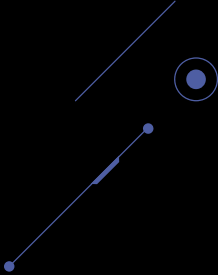
Principais funções

(define-struct galinha [x y dir])

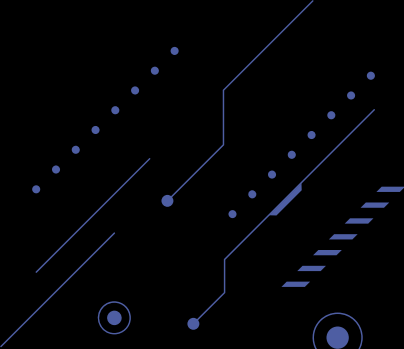
**(define (fn-para-galinha galinha)
 (... (galinha-x galinha) ; num
 (galinha-y galinha) ; num
 (galinha-dir galinha))) ; num**



(define-struct tiro [x y saiu])



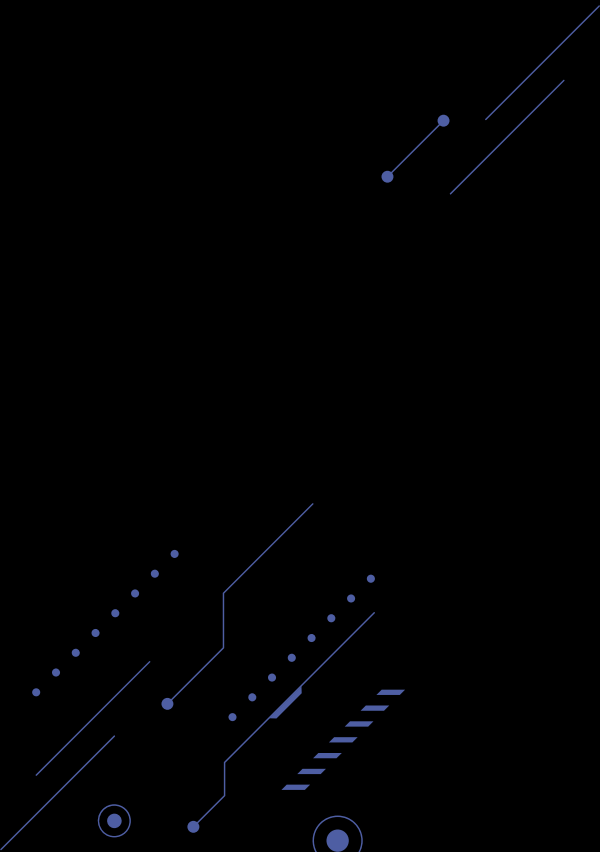
**(define (fn-para-tiro tiro)
 (... (tiro-x tiro) ; num
 (tiro-y tiro) ; num
 (tiro-saiu tiro))) ; num**





(define-struct vida [x y])

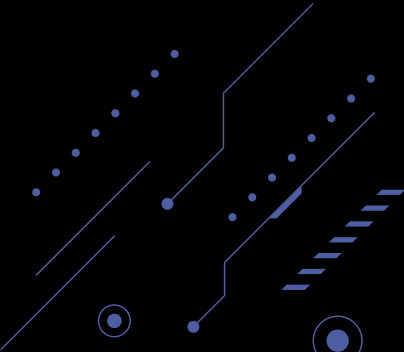
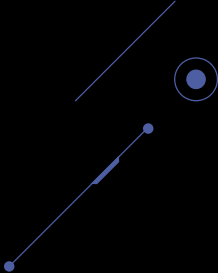
(define (fn-para-vida vida)
 (... (vida-x vida) ; num
 (vida-y vida))) ; num





(define-struct vidap [x y])

**(define (fn-para-vidap vidap)
 (... (vidap-x vidap) ; num
 (vidap-y vidap))) ; num**

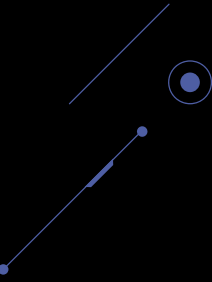
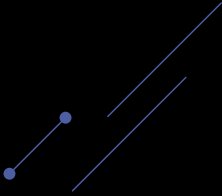
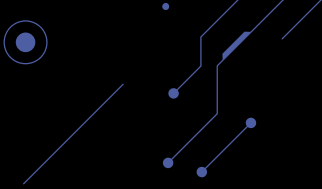


```
(define (avanca jogo)
  (if (morreu? (jogo-vidap jogo)(jogo-nave-vidap jogo))
      jogo
      (make-jogo (nave-avanca (jogo-nave jogo))
                  (ovo-avanca (jogo-ovo jogo) (jogo-galinha jogo))
                  (tiro-avanca (jogo-tiro jogo) (jogo-nave jogo))
                  (vida-avanca (jogo-vida jogo))
                  (vidap-avanca (jogo-vidap jogo) jogo)
                  (nave-vida-avanca (jogo-nave-vida jogo) jogo)
                  (nave-vidap-avanca (jogo-nave-vidap jogo) jogo)
                  (galinha-avanca (jogo-galinha jogo))))))
```

A função "avanca" é responsável por atualizar o estado do jogo. Ela verifica se algum personagem morreu (morreu?) e, se não, "avança" todos os elementos do jogo.


```
(define (desenha jogo)
  (if (morreu? (jogo-vidap jogo)(jogo-nave-vidap jogo))
      (if (retornaVidap (jogo-vidap jogo))
          (overlay (text "BOA, VOCÊ VENCEU!" 50 "green")
                    (nave-desenha (jogo-nave jogo)
                                   (ovo-desenha (jogo-ovo jogo)
                                                  (tiro-desenha (jogo-tiro jogo)
                                                                  (vidap-desenha (jogo-vidap jogo)
                                                                 (vida-desenha (jogo-vida jogo)
                                                                    (nave-vidap-desenha (jogo-nave-vidap jogo)
                                                                (nave-vida-desenha (jogo-nave-vida jogo)
                                                                    (galinha-desenha(jogo-galinha jogo))))))))))
```

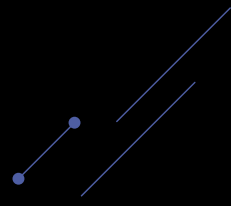
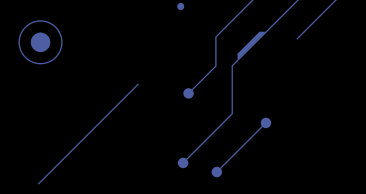
A função desenha primeiramente verifica se o jogador morreu, se sim o jogo acaba com vitória, se não o jogo todo é redesenhado.



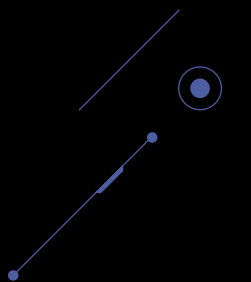
```
(define (trata-tecla jogo ke)  
  (cond  
    [(key=? ke "left") (nave-move jogo "left")]  
    [(key=? ke "right") (nave-move jogo "right")]  
    [(key=? ke "up") (nave-move jogo "up")]  
    [(key=? ke "down") (nave-move jogo "down")]  
    [else jogo]))
```

Função que trata do movimento da nave, a partir das teclas utilizadas.




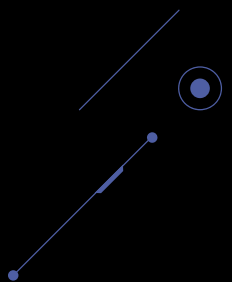
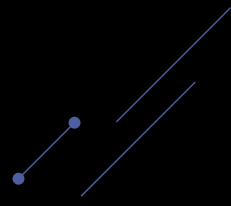
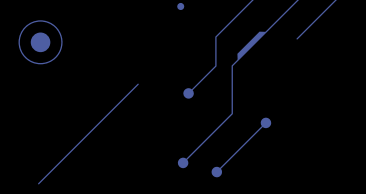


```
(define (main jogo)  
  (big-bang jogo  
    [name "chicken invader"]  
    [to-draw desenha]  
    [on-tick avanca]  
    [on-key trata-tecla]))
```




Essa função pode ser usada para inicializar o jogo, ela praticamente “dita o jogo” de acordo com o tempo, ou seja, é responsável por atualizar o jogo (chamando outras funções).





```
(define (nave-anda nave direcao)
  (if (string=? direcao "right")
      (if(> (nave-x nave)LARGURA)
          (make-nave (- (nave-x nave) LARGURA)
                      (nave-y nave))
          (make-nave (+ (nave-x nave) NAVE-VEL)
                      (nave-y nave)))
      ....))
```

Nessa função é onde ocorre o tratamento do movimento da nave, ou seja, ela recebe o comando do jogador e atualiza a posição da nave de acordo com o mesmo.





```
(define (caixa-colisao? a b)
```

```
(and
```

```
  ; borda direta de a vem antes da borda esquerda de b
```

```
  (< (caixa-x a) (+ (caixa-x b) (caixa-largura b)))
```

```
  ; borda direita de b vem antes da borda esquerda de a
```

```
  (< (caixa-x b) (+ (caixa-x a) (caixa-largura a)))
```


```
  ; borda superior de a vem antes da borda inferior de b
```

```
  (< (caixa-y a) (+ (caixa-y b) (caixa-altura b)))
```

```
  ; borda superior de b vem antes da borda inferior de a
```

```
  (< (caixa-y b) (+ (caixa-y a) (caixa-altura a))))
```

Esta função tem como objetivo verificar se existe colisão entre dois objetos "a" e "b". Ela calcula as coordenadas das bordas dos objetos e verifica se elas se sobrepõem, retornando verdadeiro ou falso com base na verificação.



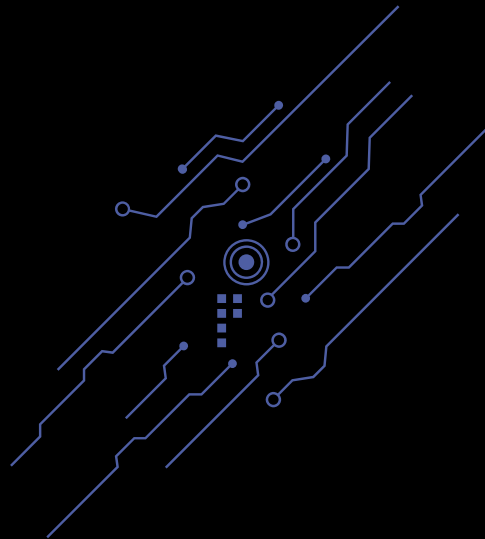
Bibliografia Utilizada

Bibliotecas usadas

- (require 2htdp/image)
- (require 2htdp/universe)
- (require 2htdp/batch-io)

Site

- <https://docs.racket-lang.org/>



MERCI !



Avez-vous des questions ?

CRÉDITS: Ce modèle de présentation a été créé par **Slidesgo**, comprenant des icônes de **Flaticon**, des infographies et des images de **Freepik**

CREDITS: This presentation template was created by **Slidesgo**, including icons by **Flaticon**, and infographics & images by **Freepik**