Como jogar Archwizard Duel

Conectando-se em uma sessão

- Um jogador deve ser o ***host*** da sessão, criando-a em um endereço de IP válido (onde haja um servidor NetGames) com o botão **"*Create Session*"**. O *host* deve ser o único usuário conectado no momento em que a sessão for criada.
- O outro jogador deve utilizar esse mesmo endereço de IP para entrar como convidado na sessão através do botão **"*Join Session*"**.

Iniciando uma partida

- Na interface de sessão, o jogador que se conectou como cliente deve esperar o início da partida, ou deixar a sessão com **"*Quit Session*"**.
- Somente o *host* pode iniciar a partida, mas somente se houver algum outro jogador conectado naquela sessão. Basta então pressionar o botão **"*Start Match*"**.

Jogando Archwizard Duel

Nesse ponto, o mapa do jogo é mostrado a ambos os jogadores, assim como os pontos de vida e magia do personagem associado ao usuário.

Existem na arena três tipos de elementos:

- **Rochas**, que são permanentemente fixas no mapa e podem ser usadas como barreiras contra feitiços.
- **Magos**: os personagens controláveis por cada jogador. **O *host* da sessão controla o mago que inicia o jogo no canto esquerdo do mapa, enquanto o outro usuário controla o que aparece no canto direito.**
- **Magias**, que são invocadas pelos magos e ao atingir um inimigo reduzem seus pontos de vida.

O ganhador de um duelo é aquele que conseguir fazer com que o mago adversário perca todos os seus pontos de vida.

Cada jogada é realizada seguindo os comandos programados pelo jogador e enviados em seu turno. **Um turno termina quando todos os comandos forem executados ou quando houver algum problema na execução do código** (se o mago ficar sem mana, por exemplo).

A linguagem se aproxima de LISP e conta com alguns mecanismos para controle de fluxo de execução, operações aritméticas, definições de variáveis, criação de procedimento e utilização de funções de primeira ordem. São eles:

```
; todo texto que vem depois de um ';'
; é tratado como comentário
```

Figure 1: prototipo

```
; definindo uma variavel x com o valor `false`
(define x false)

; aqui definimos um procedimento `foo!` que recebe 2 parametros:
; uma outra funcao sem argumentos e uma direcao
(define foo! (lambda (test dir)
  ; abaixo um uso da condicional if-then-else
  (if (test)          ; se a funcao (test) retornar algo verdadeiro...
      (begin          ; um bloco begin é executado sequencialmente
        (step!)       ; o mago da um passo à frente
        (fireball!)) ; e lança uma bola de fogo naquela direcao
      (turn! dir)))) ; caso contrário, ele se vira na direção dada

; por fim, podemos invocar `foo!` passando os parametros a seguir
(foo!
  ; o 1o argumento é uma funcao que verifica se o caminho à frente está livre
  (lambda () (not (blocked?)))
  ; e o segundo é a direcao para qual o mago pode virar
  RIGHT)

; algumas operacoes aritmeticas basicas tambem estao presentes
(+ 1 1) ; resulta em 2, como esperado
```

Uma API mágica

Seguem listados a seguir os procedimentos e valores primitivos disponíveis na linguagem de Archwizard Duel.

- `UP`: valor utilizado para representar a direção de movimentação acima.
- `DOWN`: valor utilizado para representar a direção de movimentação abaixo.
- `LEFT`: valor utilizado para representar a direção de movimentação à esquerda.
- `RIGHT`: valor utilizado para representar a direção de movimentação à direita.
- `(turn! dir)`: vira o mago para a direção desejada, devendo ser alguma dentre `UP`, `DOWN`, `LEFT` ou `RIGHT`. Custa 5 de mana.
- `(step!)`: faz com que o mago dê um passo à frente, consumindo 10 de mana.
- `(fireball!)`: lança uma magia de bola de fogo na direção à qual o mago está virado. O feitiço é dissipado após atingir algum obstáculo e causa 20 pontos de dano ao oponente. Custa 30 pontos de mana.
- `(blocked?)`: confere se o mago pode dar um passo à frente, ou se o caminho está bloqueado.

- `(mana)`: retorna a mana disponível nesse instante.
- `(health)`: : retorna a vida disponível nesse instante.
- `false`: valor booleano falso.
- `true`: valor booleano verdadeiro.
- `(not x)`: inversão lógica do argumento `x`.
- `(= a b)`: comparação de igualdade entre valores numéricos.
- `(< a b)`: comparação `a` menor que `b`.
- `(> a b)`: comparação `a` maior que `b`.
- `(<= a b)`: comparação `a` menor ou igual que `b`.
- `(>= a b)`: comparação `a` maior ou igual que `b`.
- `(+ a b)`: soma `a` e `b`.
- `(- a b)`: subtrai `b` de `a`.
- `(* a b)`: multiplica `a` e `b`.
- `(/ a b)`: divide `a` por `b`, gera um erro quando `b` é zero.
- `(round x)`: retorna o valor `x` arredondado ao inteiro mais próximo.
- `(quotient a b)`: resultado da divisão inteira de `a` por `b`, gera um erro quando `b` é zero.
- `(remainder a b)`: resto da divisão inteira de `a` por `b`, gera um erro quando `b` é zero.
- `(boolean? var)`: confere se `var` é um valor booleano.
- `(number? var)`: confere se `var` é um valor numérico.