UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
CENTRO INTERDISCIPLINAR DE NOVAS TECNOLOGIAS NA EDUCAÇÃO

CURSO DE EXTENSÃO:
Projeto e Desenvolvimento de Materiais Educacionais com Flash
Aula 5 – Interatividade e Aprendizagem

Interação e interatividade

É importante entender a diferença entre interação e interatividade. Alguns autores defendem que o termo interatividade já está com seu sentido 'esvaziado', já que se tornou lugar-comum para promover homepages, softwares e aplicações que não apresentam nada além de uma seqüência de cliques. Por isso, o termo interação é mais usado para expressar um esforço cognitivo geralmente maior do que a interatividade simplesmente reativa.

Alex Primo defende que é a capacidade interacional da Internet que pode trazer maiores contribuições à EAD, desde que o design do curso e o papel dos professores, em conjunto, primem pela interação mútua em contraste à interação reativa. Para o autor, a interação reativa pode ser facilmente implementada num contexto homem-computador; enquanto a interação mútua é aquela resultante de uma relação entre duas pessoas, num processo de "agir conjuntamente". Nas situações de interação mútua, "os participantes são definidos pelos relacionamentos, isto é, as relações afetam os seus participantes, como também seus relacionamentos futuros".

A maioria das aplicações online (cursos, jogos, objetos de aprendizagem, testes, exercícios, etc), que se definem interativos, entendem a interatividade a partir do conceito de interação reativa, onde a reação do estudante foi prevista pelos desenvolvedores da tarefa e implementada como resposta correta pelo designer do curso, de modo que se o estudante reagir de uma forma não prevista, o sistema apresenta um erro e a realização da atividade é considerada inválida ou insatisfatória.

Por outro lado, como a interação mútua não é a simples soma de traços pessoais dos interagentes, seu resultado não pode ser previsto.

Considerando-se o conceito piagetiano de construção do conhecimento, baseado em desequilíbrios seguidos de equilibrações, do ponto de vista educacional, esta parece ser a diferença mais importante entre interação mútua e reativa. Nesse sentido, um conflito age como um ponto prejudicial à interação reativa, podendo levar ao seu término; enquanto na interação mútua o conflito modifica a relação interacional, redefinindo-a.

Introdução à linguagem ActionScript visando interatividade

Variáveis

Uma variável é um recipiente que armazena informação. Essa informação pode ser, por exemplo, uma seqüência de caracteres. Isso vai depender do tipo de dado que a variável está armazenando. Uma vez que você definir uma variável em sua aplicação ela terá o mesmo nome ao longo do tempo de execução, mas seu conteúdo (valor) poderá mudar. Por exemplo, num jogo qualquer, o escore irá aumentar à medida que o usuário for fazendo mais pontos. Dessa forma, se tivermos uma variável chamada escore, ela será a mesma do início ao final do jogo, porém seu valor passará de 0 até o total de pontos que o usuário fez.

Quando temos certeza que um dado não terá seu valor alterado, o chamamos de constante. Se estamos trabalhando com o número pi, p. ex. sabemos que é um valor exato, que nunca se alterará. Dessa forma, ao invés de usar uma variável para esse valor, podemos usar uma constante. Em Actioncript 2.0, não se pode definir constantes, mas temos BACKSPACE, ENTER, SPACE e outras como propriedades da classe Key como valores constantes.

Tipos de dados

Dados referem-se a números, strings (seqüências de caracteres, como uma palavra, p.ex.) e outras informações que podem ser manipuladas no Flash. Você pode definir dados em variáveis dentro do Flash ou pode carregar dados a partir de arquivos externos, como veremos mais adiante no curso. Os tipos mais comuns de dados são:

- Strings: seqüência de caracteres, que pode incluir caracteres alfa-numéricos e sinais de pontuação;
- Números: podem ser inteiros (integer), ponto-flutuante (floating point);
- Valores booleanos: falso (false ou 0) ou verdadeiro (true ou 1), valor inicial = false;
- Nulo: sem valor, ausência de dados, valor inicial = void;

Cada tipo de dado permite tipos diferentes de operações. Por exemplo, os números podem ser somados, diminuídos, etc; as strings podem ser concatenadas (unidas, ex. "Curso" + " de " + "Flash" resulta em "Curso de Flash") e assim por diante. Dados geralmente podem ser convertidos de um tipo para outro, através do uso de funções específicas.

Os dados são armazenadas em variáveis. Você pode definir os tipos de dados quando criar suas variáveis, o que chamamos de 'declaração de variáveis'. Cada linguagem tem sua sintaxe própria e exige uma estruturação específica na declaração de variáveis.

Declaração de variáveis

No caso do ActionScript 2.0, para se declarar uma variável usamos a sintaxe estruturada pela palavra var seguida do nome da variável seguida de dois-pontos e o tipo da variável. Se você já desejar definir um valor pra essa variável, basta colocar o sinal de igual seguido do valor. Lembre-se que sempre encerramos uma linha de código com ponto-e-vírgula.

```
// sintaxe de declaração de variável: var nome_da_variavel: tipo_da_variavel = valor_atribuido_a_variavel;
var MinhaVariavelQuerida: Number = 7; // declara a variável MinhaVariavelQuerida como numérica de valor 7
trace(MinhaVariavelQuerida); // retorna 7
```

Strings são seqüências de caracteres e devem ser informadas no código através de aspas simples ou duplas. Uma forma comum de se usar strings é associá-las a variáveis. O exemplo a seguir mostra a variável Str_Software recebendo o valor "Flash8":

```
var Str_Software: String = "Flash8";
```

Strings podem ser concatenadas (unidas) através do operador (+). O ActionScript 2.0 considera os espaços no início ou final de uma string como parte da mesma. Nesse caso, teremos:

```
var Str_Saudacao: String = "Olá, " + Str_Nome;
```

Escopo de variáveis

Escopo de variáveis se refere à área na qual a variável está definida e pode ser referenciada (usada). A área em que a variável está definida pode ser dentro de uma certa linha de tempo, ou dentro de uma função, ou ainda ela pode ser global, o que significa que pode ser usada em toda a aplicação.

É muito importante entender o escopo das variáveis que você está criando pois ele não indica somente quando e onde essa variável pode ser referenciada, mas também por quanto tempo ela vai existir em determinada aplicação. Quando você define uma variável no corpo de uma função, por exemplo, ele 'pára' de existir assim que a função terminar. Dessa forma, se você tentar acessar aquela variável e ela já estiver 'deixado de existir', acontecerão erros na sua aplicação.

Existem três tipos de escopo de variáveis no ActionScript 2.0:

1. Variáveis Globais. São visíveis em qualquer local da aplicação. Para criar variáveis globais, você simplesmente coloca o identificador _global na frente do nome da variável, separados por um ponto e sem a palavra var. Você não pode definir tipo de dados nesse caso. Observe essa sintaxe:

   ```
   _global.Str_Departamento = "CINTED"; // observe que nesse caso não se usa a palavra var
   ```

2. Variáveis de Linha de Tempo. São visíveis somente para os scripts daquela linha de tempo. São declarados com o identificador var, dentro de uma linha de tempo específica. A variável só está disponível a partir do quadro onde foi declarada. Assim, se você quer que sua variável esteja disponível em toda a linha de tempo, deve declará-la no primeiro quadro.

3. Variáveis Locais. São visíveis dentro do corpo da função ou no local onde são declaradas. Atente-se ao nome que você dá a suas variáveis locais, pois se você usar um nome já usado em uma variável global, essa variável global não será acessada dentro do escopo da local. Da mesma forma, o nome de uma variável local precede uma variável de linha de tempo. Elas são declaradas também com o identificador var.

Expressões

Expressões são qualquer combinação válida de elementos do AS que resultam em um valor. Expressões têm valores; enquanto valores têm tipos. No caso dos tipos numéricos, eles podem ser manipulados usando-se operadores aritméticos de:

- adição (+): 3 + 2 resultará 5
- adição consignada (+=): +=3 somará 3 a uma variável
- subtração (-): 3 - 2 resultará 1
- subtração consignada (-=): -=3 subtrairá 3 de uma variável
- multiplicação (*): 3 * 2 resultará 6
- divisão (/): 3 / 2 resultará 1.5
- módulo (%): 3 % 2 resultará 1 // 3 / 2 = 1 inteiro + RESTO 1
- pré incremento (++): ++ 3 resultará em 4
- pós incremento (++): 3++ resultará em 3 // 4 será armazenado
- pré decremento (--): -- 3 resultará em 2
- pós decremento (++): 3-- resultará em 3 // 2 será armazenado

Operadores lógicos são usados nas expressões que comparam valores booleanos (falso ou verdadeiro) e retornam um valor booleano como resultado. Há dois operadores lógicos: AND (&&) e OR (||).

É importante saber que...

true && true = TRUE
true && false, false && true = FALSE
false && false = FALSE
true || true = TRUE
true || false, false || true = TRUE
false || false = FALSE

Vamos ver se você entendeu... Se (3>2) || (2>3) então 'diga oi para o colega da sua esquerda' senão 'diga oi para o colega da sua direita'. Quem você deve cumprimentar?

Outros operadores importantes são:

- atribuição: =
- igualdade: ==
- diferença: !=
- menor que: <
- maior que: >
- menor ou igual que: <=
- maior ou igual que: >=

Por fim, é importante ressaltarmos:

- Caso sensitivo. faz diferença entre maiúsculas e minúsculas, p.ex. str_MinhaVariavel é diferente de str_minhaVariavel;
- Sintaxe de ponto. No AS 2.0 usamos uma sintaxe de ponto para acessar propriedades ou métodos que pertencem a um objeto ou instância no palco. A expressão da sintaxe de ponto se inicia com o nome do objeto (que pode ser um clipe de filme) e termina com a propriedade ou método a ser especificado, p.ex. MeuClipeDeFilme.play(); ou MeuClipeDeFilme._x.

Condicionais e Repetições

Em programação (e na vida também!), é muito comum que nos deparemos com uma condicional e a partir do resultado dessa condição, seguimos um ou outro caminho. Por isso, os comando iterativos são tão importantes. Imagine uma situação simples, em que você tem um arquivo que calcula nota dos seus alunos:

Se a nota final for maior ou igual a 70
então o aluno passou
senão ele rodou

Fazendo um algoritmo a partir dessa situação, temos algo como:

Se (nota final >= 70)
então resultado=aprovado
senão resultado=reprovado

OU

```
if (nota final >= 70)
then (resultado=aprovado)
else (resultado=reprovado)
```

Em AS 2.0, podemos dizer que essa situação se refletiria da seguinte forma:

```
if (notafinal >= 70) {
    resultado="aprovado";
} else {
    resultado="reprovado";
}
```

No AS, a palavra-chave then (então) é omitida. Porém, da mesma forma que no algoritmo anterior, todo o bloco de instruções que estiver entre as chaves seguidas da condição será executado se a condição for verdadeira e todo o bloco de instruções que estiver entre as chaves seguidas de else será executado se a condição for falsa. Lembre-se que nem sempre é necessário colocar o else. Em algumas situações, não haverá instruções para uma condição negativa.

Algumas vezes é necessário checar mais do que uma condição. Observe o exemplo:

```
if ((notaPort>= 70) && (notaMat>= 70) && (notaCiencias>= 70)) {
    resultado="aprovado";
} else {
    resultado="reprovado";
}
```

OU

```
// Onde você está fazendo seu curso de Flash?
resposta="";
if ((resposta=="") || (resposta==null)) {
    mensagem="Você deve preencher uma resposta.";
} else if (resposta=="CINTED") {
    mensagem="Parabéns, você acertou!";
} else {
    mensagem="Infelizmente você errou...";
}
```

Assim como existem os comandos condicionais, também as estruturas de repetição são extremamente importantes na programação. Elas servem para que a gente possa checar determinada condição continuamente, ou para repetir uma ação determinadas vezes.

A repetição for deixa você repetir uma variável ou um valor específico, sendo útil quando você sabe exatamente quantas vezes você precisa repetir a série de instruções. Isso pode ser útil, p. ex., se você precisar duplicar clipes de filme no palco um certo número de vezes ou, ainda, se você precisar percorrer um array, realizando uma ação em cada item do array.

O comando for é escrito da seguinte maneira: for (init; condition; update) { instruções; }, onde init é o valor inicial, condition é a condição de repetição e update é a expressão de alteração do valor. Suponhamos que precisamos de uma repetição em que o valor inicial é 0 e o valor final é 4. A expressão de alteração será

incrementar 1 ao valor. Assim, a condição será que essa alteração aconteça até que o valor chegue a 4. Nesse caso, isso vai acontecer 5 vezes. Observe o seguinte código:

```
var i:Number = 0; // cria a variável i e declara como um número, valor 0
for (i = 0; i <5; i++) { // partindo de i=0, para i <=5, incrementa i (i++)
   trace(i); // retorna 0 1 2 3 4 distribuídos em 5 linhas
}
```

A estrutura while (enquanto) é usada para repetir uma ação enquanto uma condição existe. O while avalia uma expressão e executa o código se a expressão for verdadeira. Repetições while são úteis quando você não sabe quantas vezes você precisará repetir o bloco de instruções. Observe o exemplo:

```
var i:Number = 0; // declara i como número, valor = 0
while (i<5) { // enquanto i menor que 5
   trace(i); // retorna 0 1 2 3 4 distribuídos em 5 linhas
   i++; // incrementa i
}
```

Uma das desvantagens em usar o while ao invés do for é que com ele é mais fácil de escrever repetições infinitas. O for não roda se você omite a expressão de alteração (i++) mas o while roda mesmo sem essa informação, tornando-se uma repetição infinita.

Você pode usar o do.. while para criar o mesmo tipo de repetição que o while. Contudo, a expressão é avaliada no final do bloco de repetição, então a repetição sempre retorna ao menos um valor. As instruções são executadas somente se a condição for verdadeira. Observe o seguinte código:

```
var i: Number = 0; // declara i como número, valor = 0
do { // execute
   trace(i); // mostre o valor de i
   i++; // incrementa i
} while (i<5); // enquanto i menor que 5
```

O comando switch cria uma estrutura de ramificação para as instruções. Ele é similar à condicional if.. then já que ele testa uma condição e executa as instruções se a condição for verdadeira. Todos os switch devem conter um caso padrão. O caso padrão deve ser sempre o último caso na lista de casos do switch. Além disso, cada bloco deve conter uma instrução break como fechamento de instrução.

Observe e teste o seguinte código:

```
var sexo: String = ""; // declara sexo como string e valor=""
var msg: String = ""; // declara sexo como string e valor=""
switch (sexo) {
   case "F": msg="feminino"
      break
   case "M": msg="masculino"
      break
   default: msg="não informado"
      break
}
trace(sexo);
```

A maioria das linguagens de programação e, nesse caso, a ActionScript 2.0, oferecem funções, que são rotinas prontas para serem reutilizadas a partir da inserção de comandos curtos de código. No caso do AS, play(); e stop(); são funções!

No AS 2.0, por ser uma linguagem OO, as funções estão sendo transformadas em métodos. Para conhecer todos os métodos de uma classe, vá no Painel de Ações, clique em (+) >> ActionScript 2.0 Classes >>. Não se preocupe, com o tempo você vai se familiarizando com as funções/métodos da AS!

Introdução à programação orientada a objetos (POO)

AS 2.0 é uma linguagem orientada a objetos e, por isso, é baseada no conceito de classes e instâncias (cópias). Uma classe define todas as propriedades que distinguem uma série de objetos. Por exemplo, uma classe Usuário representa um grupo de usuário que está usando uma aplicação. Então, se você faz um instanciamento dessa classe, você terá um dos seus usuários individuais, ou seja, um dos seus membros. Esse instanciamento produz uma instância da classe Usuário e essa instância tem todas as propriedades da classe Usuário (nome, departamento).

As classes podem ser consideradas como templates (modelos) que você pode criar para definir um novo tipo de objeto. Por exemplo, se você precisa trabalhar com o tipo de dado Usuário, então você define a classe Usuário. Isso definirá também o objeto Usuário. Esse objeto terá propriedades, nesse caso, nome e departamento; e também métodos, como cadastrar(), editar(), apagar(). Para definir uma classe, como a classe Usuário, você deve usar a palavra-chave class em um arquivo externo de script. Você pode criar um arquivo externo de script em File > New > ActionScript File.

Como sempre, AS 2.0 oferece diversos conceitos de POO poderosos e palavras-chave (tais como class, interface e package), encontradas em outras linguagens OO, tais como Java. Esse tipo de linguagem de programação deixa você construir estruturas que são reusáveis, escaláveis, robustas e preserváveis. Por isso, ela pode diminuir o tempo de desenvolvimento. Você pode usar a AS 2.0 para criar objetos e estabelecer herança.

Em linguagens POO, uma classe define uma categoria de objeto. Uma classe descreve propriedades (dados) e métodos (comportamentos) para um objeto. Você pode escrever uma classe personalizada em um arquivo externo de AS (terminação .as) e importá-la para sua aplicação quando você compilar o arquivo FLA.

As classes também podem ser muitos úteis quando você constrói aplicações Flash complexas porque você pode colocar toda a complexidade da aplicação no arquivo externo de classe. Quando você realoca a complexidade em uma classe personalizada, você não só faz com que o código fique mais simples e reusável, mas também 'esconde' alguns de seus métodos e propriedades, o que pode prevenir o acesso a informação sigilosa ou a alteração indevida do código.

Alguns conceitos são fundamentais para se entender as linguagens de POO. São eles:

- Objetos. São os 'membros integrantes' das classes, que têm propriedades e comportamentos (métodos).
- Instâncias. São exemplos de um objeto, ou seja uma 'cópia' daquele objeto com a qual se está trabalhando.
- Herança. Um dos benefícios da OO é que se você criar um subclasse e ela herdará as características da classe. Usando subclasses é mais fácil reusar códigos similares entre várias classes. A subclasse pode definir novos métodos e propriedades específicos, ou mesmo redefinir os já existentes.
- Encapsulamento. Num código OO bem estruturado os objetos são 'caixas' que contêm (ou encapsulam) funcionalidades. Um programador deve conseguir interagir com um objeto somente por conhecer suas

propriedades, métodos e eventos, sem necessidade de saber os detalhes de sua implementação. Isso permite ao programador pensar em níveis mais altos de abstração e contruir sistemas mais complexos.
- Polimorfismo. A POO permite que você expresse diferenças entre indivíduos de uma classe, usando a técnica do polimorfismo, pela qual as classes podem anular métodos de suas superclasses e definir implementações especializadas desses métodos.

Além dos elementos e construtores principais da linguagem, bem como dos tipos de dados primitivos, a AS 2.0 também oferece muitas classes embutidas (pré-definidas). Essas classes determinam uma variedade de características e funcionalidade para os scripts. Por exemplo, você pode usar a classe Math para realizar equações em suas aplicações ou, ainda, a classe BitmapData para criar animações.

Top-level classes são encontradas no painel Actions > ActionScript 2.0 Classes > Core. Essas classes são aquelas comuns às linguagens de POO. As demais classes encontradas em Actions > ActionScript 2.0 Classes são aquelas específicas do Flash, que servem pra controlar o filme Flash, elementos de mídia, XML, dentre outros.

Criação de testes e exercícios interativos

Diversos softwares educacionais, dentre eles o próprio Flash, oferecem exercícios pré-prontos, para serem utilizados em treinamentos. Apesar do caráter marcadamente comportamentalista desses exercícios, eles podem ser úteis em algumas situações. Além disso, se o professor for capaz de desenvolver seus próprios exercícios, será mais fácil adaptá-los a uma aula mais atual (entenda-se mais construtivista, mais interacionista, menos centrada no professor, etc.).

Os quatro principais tipos de exercícios normalmente oferecidos por tais softwares são:

- múltipla escolha;
- associação (ou ligar os elementos);
- verdadeiro ou falso;
- questionário com resposta simples (ou preencher espaços em branco).

1 - Múltipla Escolha

O exercício de múltipla escolha é muito simples de ser implementado no Flash. Se você está acompanhando o conteúdo até agora, já tem condições de desenvolvê-lo. Nesse caso, os conhecimentos necessários são:

- Criar texto estático e dinâmico
- Inserir botões com ações
- Escrever rotinas condicionais

2 - Questionário (ou preencher espaços)

Os exerícios do tipo questionário são aqueles em que há uma questão que possibilite o usuário respondê-la com uma única palavra, de maneira que o resultado já pode ser avaliado pela aplicação Flash. Esse tipo de exercício também é bem simples e já pode ser implementado por você. Os conhecimentos necessários são:

- Criar texto estático, dinâmico e de entrada
- Inserir botões com ações
- Escrever rotinas condicionais

3 - Associação

O exercício de associação é aquele em que são oferecidos dois tipos de elementos, que devem ser organizados em pares pelo aluno, de acordo com algum critério definido na atividade. Ele pode ser implementado de diversas formas, mas no nosso exemplo vamos ver como criar um exercício de associação em que seja necessário arrastar um elemento em direção ao seu par.

Para esse tipo de exercício, vamos precisar dois diferentes tipos de CF: (1) um clipe de filme que será arrastado e, para isso, precisará de um botão transparente e (2) outro que será o alvo do CF arrastado. Esses dois CF devem ter seus nomes instanciados, no painel de Properties (Propriedades).

Vamos ver quais são os passos para criar uma associação entre dois elementos:

1 - Crie dois clipes de filme (Insert > Symbol > MovieClip) que serão os elementos associados, por exemplo, uma imagem e uma palavra;
2 - Crie um botão em Insert > Symbol > Button. Deixe os quadros dos estados up, over e down vazios e defina apenas a área clicável do botão. Esse tipo de botão é chamado de transparente ou invisível. No botão transparente, insira o seguinte código:

```
on (press) {
   this.startDrag (true);
  }
on (release) {
   this.stopDrag ();
  }
```

3 - Teste para ver se o CF que contém o botão já está arrastável.
4 - Agora é necessário fazer uma verificação quando o CF arrastável é largado sobre o CF alvo. Para isso, a seguinte rotina deve ser incluída no código:

```
on (press) {
   this.startDrag (true);
  }
on (release) {
   this.stopDrag ();
   if (_root.CFalvo.hitTest (this) {
      mensagem="Muito bem, você fez a associação correta!";
   } else {
      mensagem="Puxa que pena! Essa não é a associação correta.";
   }
  }
```

No código acima, CFalvo é o nome de instância do CF que serve de alvo e CFarrastavel é o nome de instância do CF que é arrastado. A propriedade _root. na frente do nome do CF serve para avisar pro Flash que esse CF está no filme principal, pois como o código está sendo colocado dentro do botão, que está dentro do CF arrastável, o Flash iria procurar por esse nome de instância somente dentro do CF arrastável. Se você procurou o nome de instância do CFalvo no alvo do painel Ações (Actions), não precisará se preocupar com isso, pois o Flash coloca automaticamente o caminho correto ;-)

4 - Verdadeiro ou Falso

O exercício de falso ou verdadeiro é aquele em que o aluno tem que recorrer a seu conhecimento, ou a um texto, vídeo, áudio, simulação ou animação, e a partir disso responder se uma sentença é verdadeira ou falsa. Este tipo de exercício também pode ser implementado de várias formas, mas no exemplo vamos criar botões personalizados para as opções V e F. Esses botões têm um clipe de filme para indicar se a opção está ou não marcada. Esse CF inicia invisível e é visibilizado ao clique na opção.

Nesse tipo de exercício, você vai precisar de: (1) um clipe de filme que será a marcação da opção (um xis, por exemplo, como se marcam as provas de vestibular); (2) um botão que deverá ser clicado para escolha da resposta V ou F.

Vamos então ao passo-a-passo para criação desse tipo de exercício:

1 - Crie um botão (Insert > New Symbol > Button) que represente a caixa de opção e salve-o como botaoV.
2 - Duplique o botão criado (Na biblioteca, clique sobre ele com o botão direito do mouse e escolha Duplicate) e salve-o como botaoF.
3 - Coloque os botões criados no palco.
4 - Crie um clipe de filme (Insert > New Symbol > MovieClip) que represente um X. Coloque duas instâncias desse CF no palco e dê os nomes de instância escolheuV e escolheuF para eles. Essas instâncias devem ficar em uma camada abaixo da camada onde estão os botões.
5 - No botão botaoV, coloque o seguinte código:

```
on (press) {
   escolheuV._visible = true;
   escolheuF._visible = false;
}
```

6 - No botão botaoF, coloque o seguinte código:

```
on (press) {
   escolheuF._visible = true;
   escolheuV._visible = false;
}
```

7 - Agora, é preciso que, ao iniciar o filme, os dois CF que mostram a marca do X estejam invisíveis. Para isso, coloque o seguinte código em cada um deles:

```
onClipEvent (load) {
   this._visible=false;
}
```

8 - Você ainda pode inserir uma rotina de testagem em cada um dos botões, para verificar se o aluno respondeu corretamente ou não.

Créditos

Universidade Federal do Rio Grande do Sul
Centro Interdisciplinar de Novas Tecnologias na Educação

Profa. Dra. Liane Margarida Rockembach Tarouco
Coordenação

Bárbara Ávila
Letícia Coelho Roland
Desenvolvimento