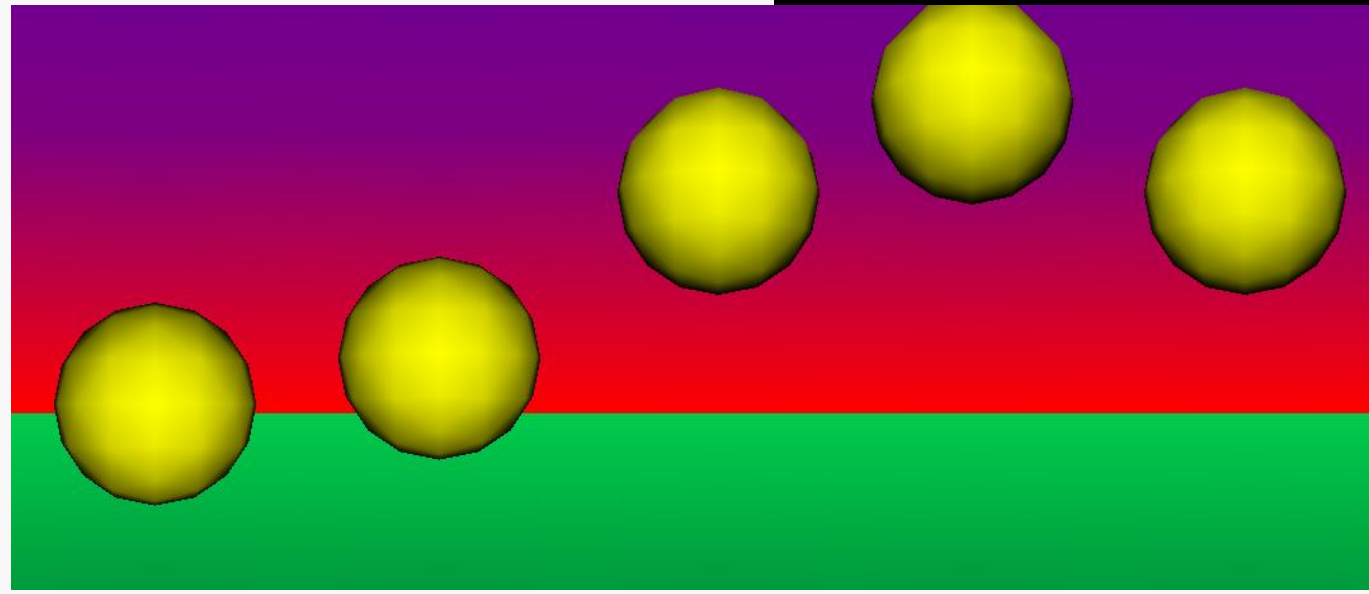
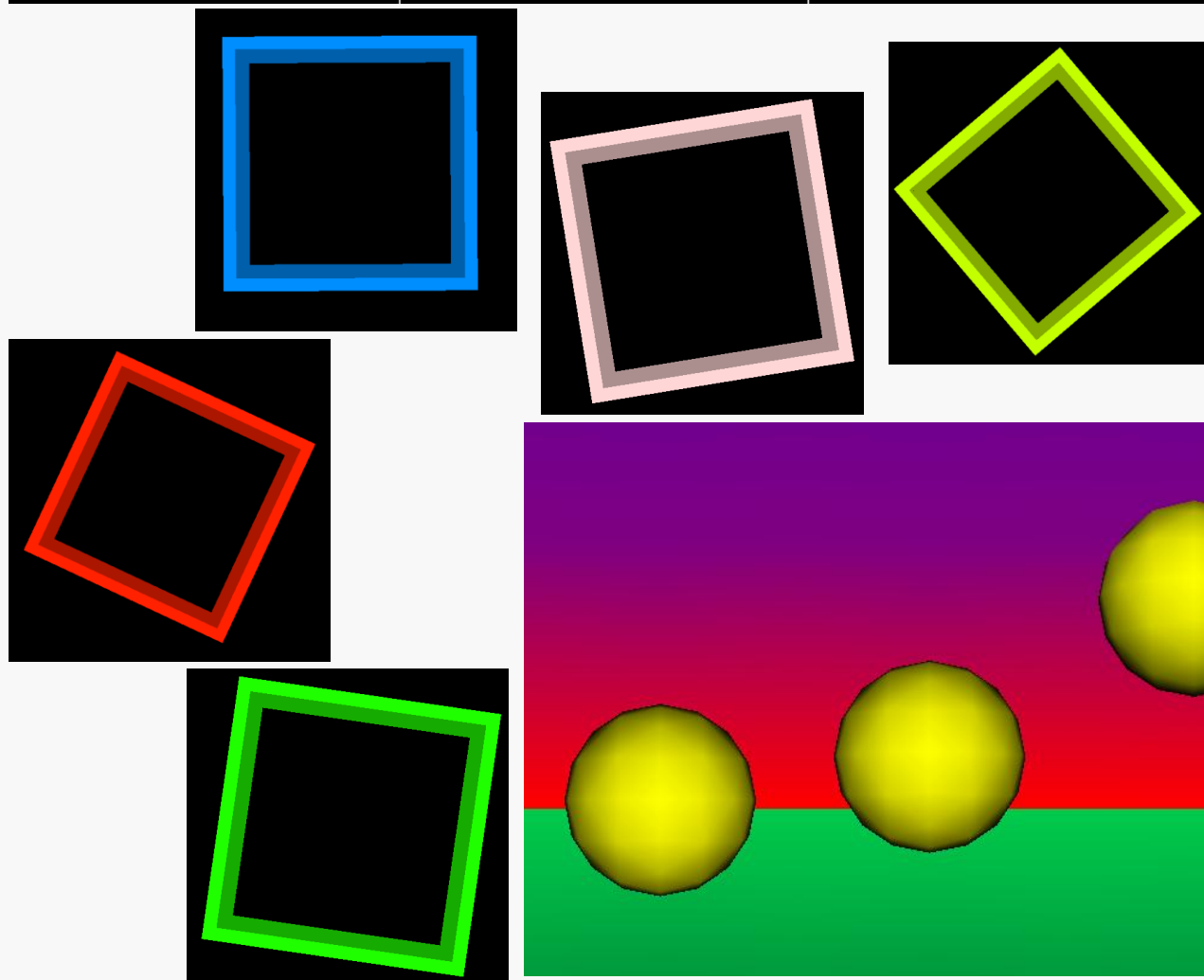
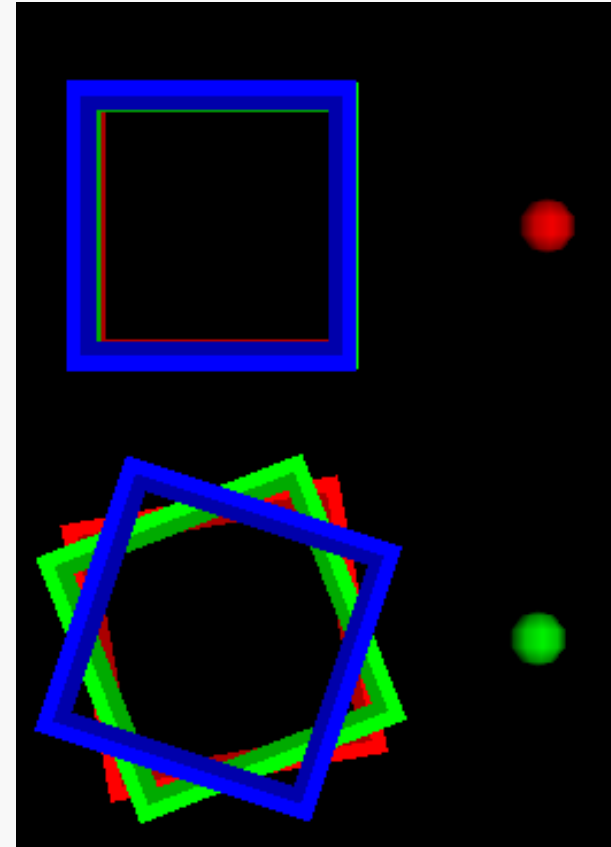
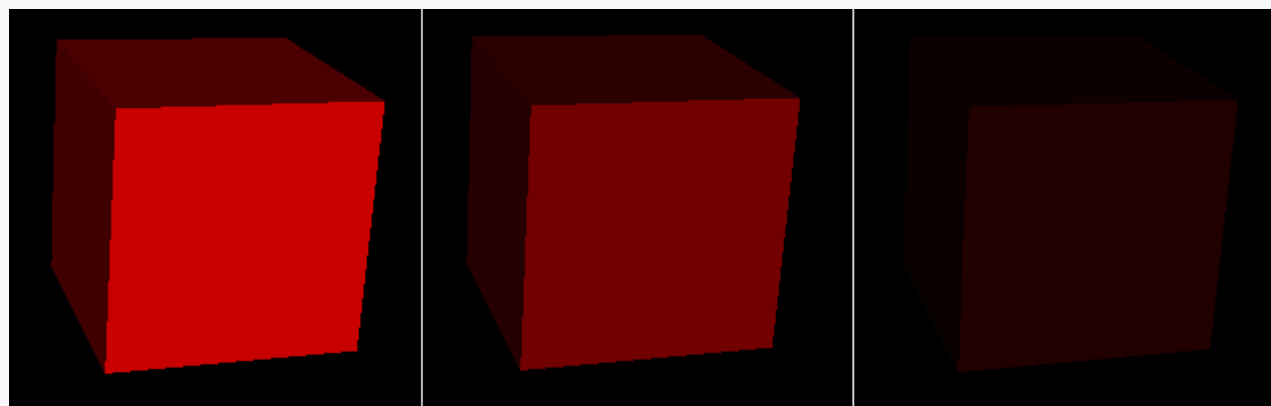

Animacão no VRML

Computacão Gráfica

Baseado no tutorial “Introduction to VRML 97” de David R. Nadeau



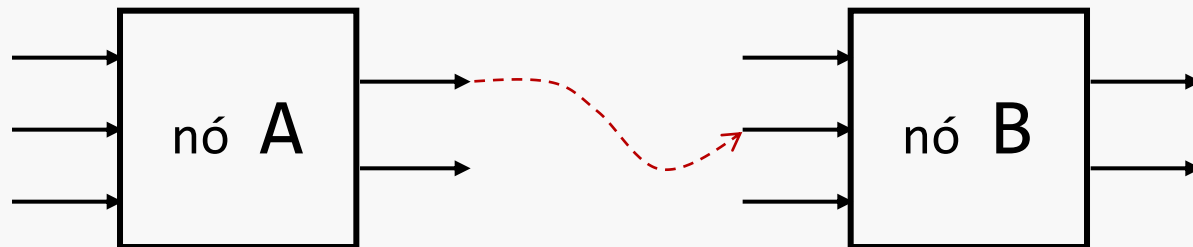


Agenda

- Circuitos de animação
 - Eventos e Rotas
- Sensores
- Interpoladores
- Utilização de Java e JavaScript em VRML

Circuitos de animação

- Quase todos os nós VRML podem pertencer a um circuito de animação
- Podem **receber** eventos
- Podem **gerar** eventos
- É necessário definir ligações entre nós
- Um evento é uma mensagem enviada entre dois nós
 - enviar uma posição (para mover o objecto)
 - enviar um tempo (quando é que ... aconteceu)

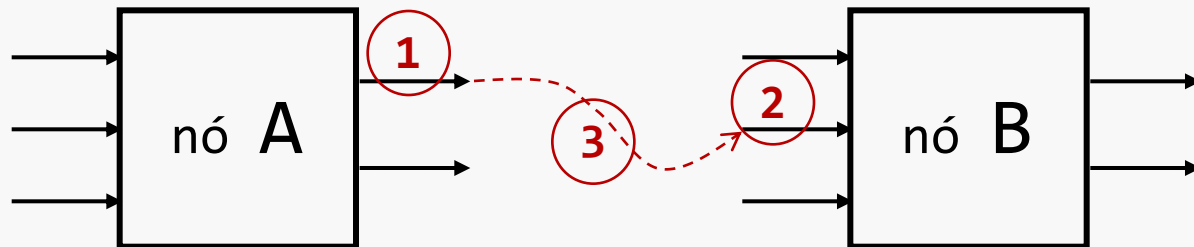


Exemplos

- Para rodar uma forma
 - Ligar um nó que envie **eventos de rotação** para o **campo rotation** do nó Transform
- Para fazer uma forma piscar
 - Ligar um nó que envie **eventos de cor** para o **campo diffuseColor** do nó Material

Definição de circuito de animação

- São necessários três campos para definir um circuito de animação
 1. Um nó que é **gerador de eventos**
 - O nó tem que ter um nome (usar *keyword* DEF)
 2. Um nó que é **consumidor de eventos**
 - O nó tem que ter um nome (usar *keyword* DEF)
 3. Criar **ligação** entre os dois nós (ROUTE)



Eventos

- Todos os nós têm
 - **campos** (field), eventos de **entrada** (eventIn) e de **saída** (eventOut)
 - Um `exposedField` é um atalho para a definição de todos estes elementos

Exemplos de *inputs*:

Nó Transform

`set_translation`
`set_rotation`
`set_scale`

Nó Material

`set_diffuseColor`
`set_emissiveColor`
`set_transparency`

Exemplos de *outputs*:

Nó TimeSensor

`time` que envia valores de tempo

Nó OrientationInterpolator

`value_changed` que envia valores de rotação

Nó PositionInterpolator

`Value_changed` que envia valores de posição (translação)

Convenção de nomes

- Quase todos os nós VRML têm `exposedFields`
- Se o nome do `exposedField` for AAA
 - `set_AAA` é o nome do evento de entrada (`eventIn`)
 - `AAA_changed` é o nome do evento de saída (`eventOut`)
 - Estes prefixo e sufixo são opcionais mas devem ser utilizados (clareza)
- Exemplo:
- O nó `Transformation` tem
 - Campo `rotation`
 - Evento de entrada `set_rotation`
 - Evento de saída `rotation_changed`

Ligações (ROUTE)

- O nó ROUTE liga dois nós através dos seus nomes (DEF) e respectivos eventos de saída (eventOut) e entrada (eventIn)

- Exemplo:

ROUTE MySender.rotation_changed **TO** MyReceiver.set_rotation

- Notas:
 - O tipo de dados dos eventos tem que ser compatível
 - Atenção que as *keywords* ROUTE e TO são em maiúsculas

Recordando o sistema de tipos do VRML

- O nome dos tipos em VRML seguem a seguinte regra
 - **1º carácter: S** – valor simples; **M** – valor múltiplo
 - **2º carácter:** é sempre um **F** (de *field*)
 - **Restantes:** nome do tipo de dados

Tipo de dados	Significado
SFBool	Boolean, true or false value
SFColor, MFColor	RGB color value
SFFloat, MFFloat	Floating point value
SFImage	Image value
SFInt32, MFInt32	Integer value
SFNode, MFNode	Node value
SFRotation, MFRotation	Rotation value
SFString, MFString	Text string value
SFTime	Time value
SFVec2f, MFVec2f	XY floating point value
SFVec3f, MFVec3f	XYZ floating point value

Sensores

- Sensores são nós capazes de **captar interacção** do utilizador e **transformá-la num evento**
- Existem dois tipos de sensores
 - Ambientais
 - TimeSensor
 - VisibilitySensor
 - ProximitySensor
 - Collision
 - Dispositivos de entrada (*mouse*)
 - TouchSensor
 - SphereSensor
 - CylinderSensor
 - PlaneSensor

TouchSensor

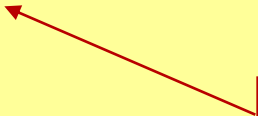
```
TouchSensor {  
  exposedField SFBool    enabled TRUE          # activo/inactivo  
  eventOut      SFVect3f  hitpoint_changed  
  eventOut      SFVect3f  hitnormal_changed  
  eventOut      SFVect3f  hitTexCoord_changed  
  eventOut      SFBool    isActive    # o sensor está activo?  
  eventOut      SFBool    isOver      # o mouse está sobre o objecto?  
  eventOut      SFTIME    touchTime   # instante de tempo associado ao toque  
}
```

- Utilizado, normalmente, para iniciar determinada acção através do evento `touchTime`
 - Começar um som
 - Colocar cronómetro em execução

Exemplo 1

- Tocar um som **quando se toca num objecto**
- Usar o nó TouchSensor em conjunto com o nó Sound

```
Group {  
  children [  
    Shape {  
      geometry Box { }  
      appearance Appearance { material Material { diffuseColor 1 0 0 } }  
    }  
    DEF TS TouchSensor { } # gerar notificações ao clique na Box  
  ]  
}  
  
Sound {  
  DEF Som AudioClip {  
    url "explosion.wav"  
  }  
}  
  
ROUTE TS.touchTime TO Som.startTime
```



Atenção que o TouchSensor
tem que ser do mesmo grupo
que o nó no qual queremos
cliquear.

TimeSensor (cronómetro)

```
TimeSensor {  
  exposedField SFBool   enabled      TRUE    # activo/inactivo  
  exposedField SFTime   cycleInterval 1      # nº de segundo de um ciclo  
  exposedField SFBool   loop        FALSE   # vai contar várias vezes?  
  exposedField SFTime   startTime    0      # tempo de início do contagem  
  exposedField SFTime   stopTime     0      # tempo de fim da contagem  
  eventOut      SFTime   cycleTime  
  eventOut      SFFloat  fraction_changed  
  eventOut      SFTime   time  
  eventOut      SFBool   isActive  
}
```

- Campos:
 - **cycleInterval**: tempo (segundos) em que a contagem é reiniciada
 - **startTime**: tempo em que começa a contagem
 - **stopTime**: tempo em que termina a contagem
- Eventos:
 - **cycleTime**: envia evento por intervalo de tempo
 - **fractionChanged**: gera eventos (*float*) com a fracção do tempo em que o contador se encontra
 - **time**: gerado ao mesmo tempo que o evento **fractionChanged** mas com o valor absoluto do tempo decorrido

Exemplo 2

- Tocar um som **a cada 3 segundos**
- Usar o nó TimeSensor em conjunto com o nó Sound

```
DEF TS TimeSensor {  
  cycleInterval 3 # é para gerar notificações de 3 em 3 segundos  
  loop TRUE # é para contar várias vezes  
}
```

```
Sound {  
  DEF Som AudioClip {  
    url "explosion.wav"  
  }  
}
```

De 3 em 3 segundos é gerado o evento **cycleTime**

```
ROUTE TS.cycleTime TO Som.startTime
```

Ao receber o evento de entrada **startTime**, o nó AudioClip coloca em execução o som indicado

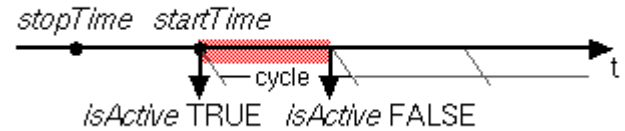
Mais sobre o TimeSensor...

- Controlamos o tempo de início e de fim
- Gera eventos de tempo enquanto está em execução
- Importante para animar outros nós
- Suporta geração de tempos **absolutos** e **relativos**
 - Absolutos
 - Medido em segundos deste as 00h00m de 1 de Janeiro de 1970
 - Utilizado para gerar eventos em determinada altura
 - Relativos
 - Gera valores entre 0.0 (início) e 1.0 (fim de ciclo)
 - O número de segundos entre 0.0 e 1.0 controlado pelo `cycleInterval`
 - Interpola os valores intermédios
 - para gerar o evento `fraction_changed`

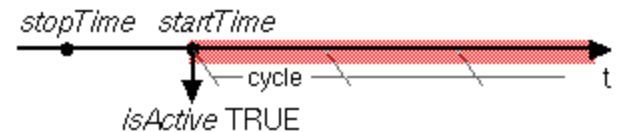
Usando o TimeSensor

- Executar um ciclo e depois parar
 - loop FALSE
 - startTime >= stopTime
 - o stopTime é ignorado
- Como criar um *timer* contínuo
 - loop TRUE
 - startTime >= stopTime
 - o stopTime é ignorado
- executar até ao stopTime
 - loop TRUE
 - startTime < stopTime
- Parar em determinado instante
 - set_stopTime = t_s
 - set_loop = FALSE

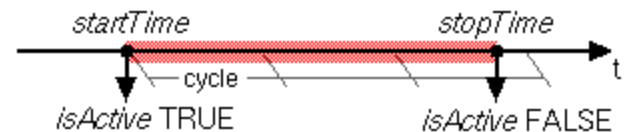
startTime >= stopTime, loop FALSE:



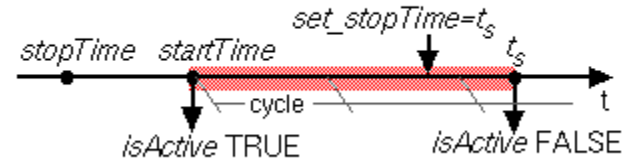
startTime >= stopTime, loop TRUE:



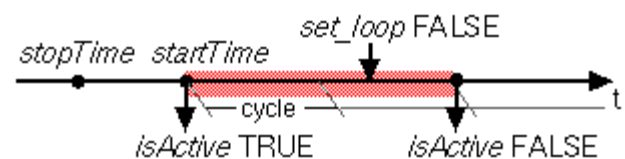
startTime < stopTime, loop TRUE:



set_stopTime, loop TRUE:



loop TRUE, set_loop FALSE:



Exemplo 3: caixa a ficar transparente...

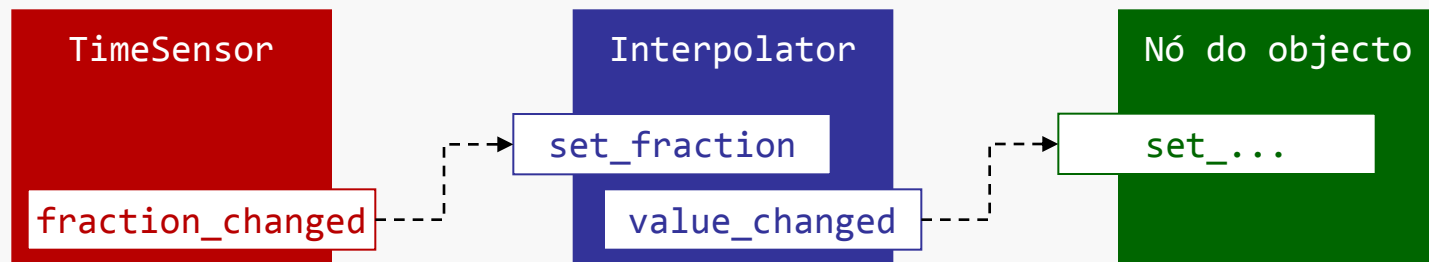
```
Group {
  children [
    Shape {
      geometry Box { }
      appearance Appearance {
        material DEF Mat Material { diffuseColor 1 0 0 }
      }
    }
    DEF Touch TouchSensor { }
  ]
}

DEF Timer TimeSensor {
  cycleInterval 4.0
  loop          FALSE
  startTime     0.0
  stopTime      0.0
}

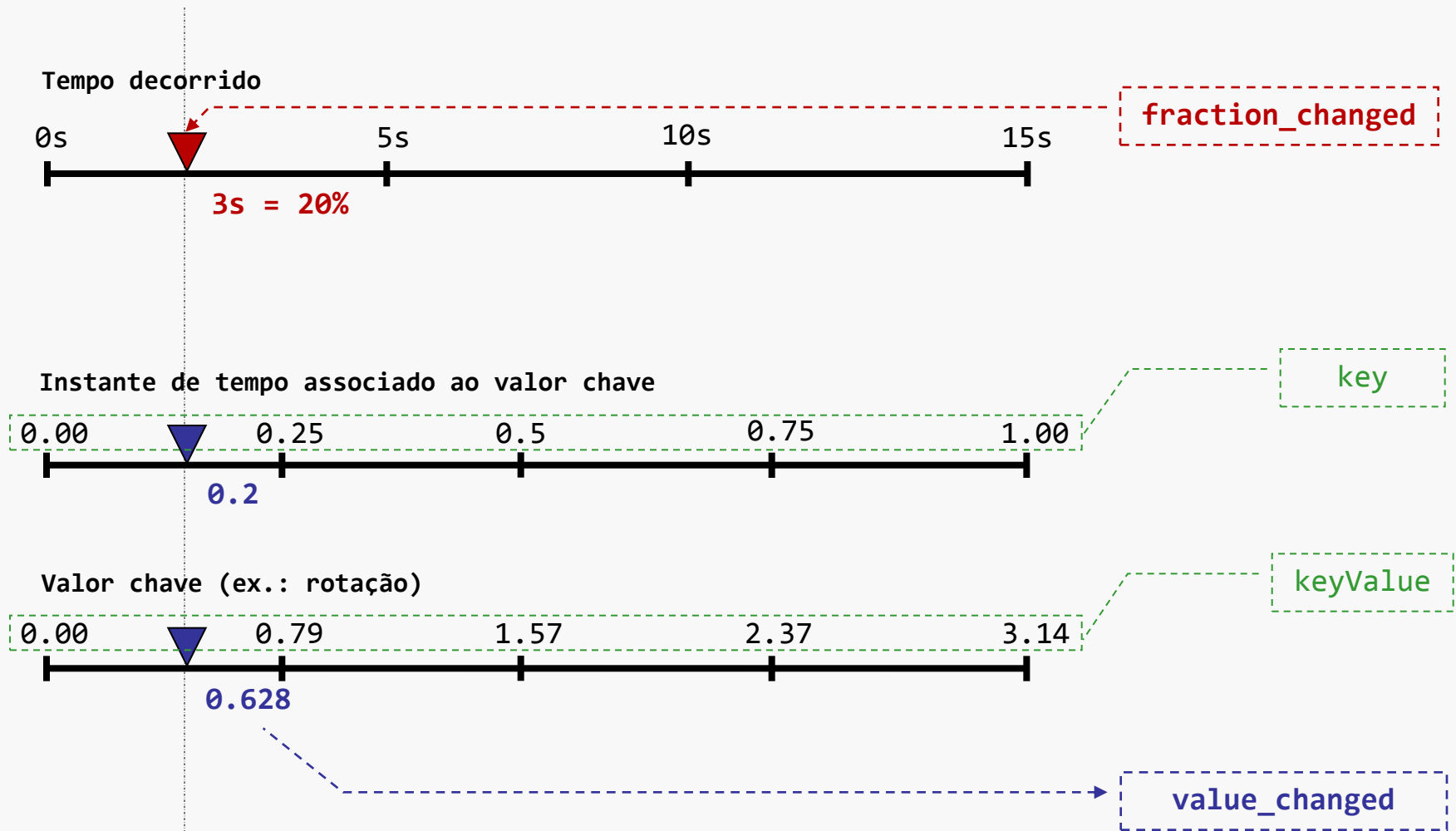
ROUTE Touch.touchTime TO Timer.startTime
ROUTE Timer.fraction_changed TO Mat.transparency
```

Como converter tempo em posições?

- Para animar a posição de uma forma é necessário
 - Uma lista de **posições chave** de determinado movimento
 - O **instante de tempo** associado a cada uma das posições chave
- Um nó **Interpolador de posições** converte um **evento de tempo de entrada** num **evento de posição de saída**
 - Para os tempos que estão entre posições chave, a posição é interpolada



Interpolação: funcionamento



Interpoladores existentes no VRML

- Cor: `ColorInterpolator`
- Posição: `PositionInterpolator`
- Rotação: `OrientationInterpolator`
- Valores escalares: `ScalarInterpolator`
- Coordenadas: `CoordinateInterpolator`
- Normais: `NormalInterpolator`

- Eventos dos interpoladores
 - Evento de **entrada**: `set_fraction`
 - Evento de **saída**: `value_changed`

Exemplo 4: bola a saltar 4m de 5 em 5secs...

```
DEF Bola Transform {  
  children Shape {  
    geometry Sphere { }  
    appearance Appearance {  
      material Material { diffuseColor 1 1 0 }  
    }  
  }  
}
```

```
DEF PosInterpolator PositionInterpolator {  
  key      [0.0    0.25   0.5    0.75   1.0]  
  keyValue [0 0 0, 0 2 0, 0 4 0, 0 2 0, 0 0 0]  
}
```

```
DEF Timer TimeSensor {  
  cycleInterval 5.0  
  loop          TRUE  
}
```

Atenção:

Este valor deve ser **igual** ao primeiro de forma a ter **continuidade** no movimento da bola

```
ROUTE Timer.fraction_changed TO PosInterpolator.set_fraction  
ROUTE PosInterpolator.value_changed TO Bola.translation
```

Exemplo 5: moldura a rodar e mudar de cor

```
DEF FrameTransform Transform {
  children [ DEF FrameInstance Frame { color 1 1 1 }
             DEF TouchStart TouchSensor { } ]}]

DEF Timer1 TimeSensor {
  cycleInterval 5 loop TRUE
}

DEF RotIInterpolator OrientationInterpolator {
  key      [0.0    0.2    0.4    0.6    0.8    1.0]
  keyValue [0 0 1 0, 0 0 1 1.26, 0 0 1 2.51, 0 0 1 3.77, 0 0 1 5.03, 0 0 1 6.28]
}

DEF ColInterpolator ColorInterpolator {
  key      [0.00  0.16  0.32  0.50  0.68  0.84  1.00]
  keyValue [1 1 1, 1 0 0, 1 1 0, 0 1 0, 0 1 1, 0 0 1, 1 1 1]
}

ROUTE TouchStart.touchTime TO Timer1.startTime
ROUTE Timer1.fraction_changed TO RotIInterpolator.set_fraction
ROUTE Timer1.fraction_changed TO ColInterpolator.set_fraction
ROUTE RotIInterpolator.value_changed TO FrameTransform.rotation
ROUTE ColInterpolator.value_changed TO FrameInstance.set_color
```

Nota:
Estes dois valores não têm o mesmo valor mas representam a mesma rotação: $0 \text{ rad} = 2\pi \text{ rad}$

Nota: Campo color definido como exposedField em Frame

Agenda

- Circuitos de animação
 - Eventos e Rotas
- Sensores
- Interpoladores
- Utilização de Java e JavaScript em VRML

E animações mais complicadas?...

- Usar nó Script
 - Realizar animações mais complexas: gravidade, etc...
 - Algoritmos com formas: *fractals*
 - Ambientes colaborativos: jogos
 - Elementos de UI
- Utiliza-se uma linguagem imperativa
 - Java
 - JavaScript (ECMAScript)

Nó Script: sintaxe

- **Código** definido através da propriedade `url` do nó `Script`

```
DEF ScriptEmJavaFicheiro Script {  
    url "bouncer.class"  
}  
DEF ScriptEmJavascriptFicheiro Script {  
    url "bouncer.js"  
}  
DEF ScriptEmJavascriptInline Script {  
    url "javascript: ..."  
}
```

- **Interface** definida com as *keywords* `field`, `eventIn` e `eventOut`

```
DEF Bouncer Script {  
    field      SFFloat    bounceHeight 3.0    # valor por omissão: 3.0  
    eventIn    SFFloat    set_fraction      # evento de entrada (tempo)  
    eventOut   SFVec3f    value_changed      # evento de saída   (posição)  
}
```


Nó Script e o JavaScript

- Na definição do código JavaScript é, normalmente, utilizada a notação *inline*
 - url “javascript: ...”
- O código é um conjunto de funções
 - para **tratamento dos eventos** de entrada
 - **Utilitárias**
 - outras com **significado pré definido**
 - A função **initialize()**, opcionalmente definida, é chamada quando o nó é carregado
 - quando o *browser* carrega a cena
 - Quando o nó é adicionado à cena
 - A função **shutdown()**, opcionalmente definida, é chamada quando o nó é descarregado
 - Quando o *browser* carrega uma nova cena
 - Quando o nó é removido da cena

Nó Script e o JavaScript: eventos

- Por cada evento de entrada (**eventIn**) é necessário definir uma função
- Função recebe dois parâmetros:
 - um valor
 - o tempo no qual ocorreu o evento (*timestamp*)

```
DEF Bouncer Script {  
  field      SFFloat  bounceHeight 3.0      # valor por omissão: 3.0  
  eventIn    SFFloat  set_fraction        # evento de entrada (tempo)  
  eventOut   SFVec3f  value_changed       # evento de saída   (posição)  
  
  url "javascript:  
    function set_fraction( frac, timestamp ) {  
      ...  
    }  
  
  "  
}
```



Nó Script e o JavaScript: eventos

- Os campos (**field**) e eventos de saída (**eventOut**) são variáveis JavaScript
 - Valores múltiplos são *arrays*

```
DEF Bouncer Script {
  field      SFFloat  bounceHeight 3.0    # valor por omissão: 3.0
  eventIn    SFFloat  set_fraction      # evento de entrada (tempo)
  eventOut   SFVec3f  value_changed      # evento de saída (posição)

  url "javascript:

    function set_fraction( frac, timestamp ) {
      y = 4.0 * bounceHeight * frac * (1.0 - frac);
      value_changed[0] = 0.0; // X
      value_changed[1] = y;   // Y
      value_changed[2] = 0.0; // Z
    }

  "
}
```

Exemplo 6: bola a saltar (com gravidade)

```
DEF BallTransform Transform {
  children Shape {
    geometry Sphere {}
    appearance Appearance { material Material { diffuseColor 1 1 0 } }
  }
}

DEF Bouncer Script {
  field      SFFloat  bounceHeight 4.0    # valor por omissão: 4.0
  eventIn    SFFloat  set_fraction      # evento de entrada (tempo)
  eventOut   SFVec3f  value_changed      # evento de saída   (posição)
  url "javascript:
    function set_fraction( frac, timestamp ) {
      y = 4.0 * bounceHeight * frac * (1.0 - frac);
      value_changed[0] = 0.0;  // X
      value_changed[1] = y;    // Y
      value_changed[2] = 0.0;  // Z
    }
  "
}

DEF Timer1 TimeSensor { cycleInterval 2 loop TRUE }

ROUTE Timer1.fraction_changed TO Bouncer.set_fraction
ROUTE Bouncer.value_changed TO BallTransform.set_translation
```

Nó Script: construindo interfaces

- O nó Script pode ser utilizado para criar elementos de interface com o utilizador (UI)
 - Botões
 - Caixas de texto
 - Mensagens de aviso

UI: Exemplo de botão ON/OFF

```
DEF ToogleButton Script {  
  field      SFBool on TRUE      # estado do botão ON/OFF  
  eventIn    SFBool set_active  # muda o estado do botão (ao receber TRUE)  
  eventOut   SFBool on_changed  # notifica alteração de estado  
  
  url "javascript:  
  
    function set_active( set, timestamp ) {  
      if(set == FALSE) return;  // apenas muda de estado no TRUE  
      // Alterar o estado  
      if(on == TRUE)    on = FALSE;  
      else              on = TRUE;  
      on_changed = on;  // gera o evento de saída com o estado do botão  
    }"  
  
}
```

- Pode ser usado em conjunto com um TouchSensor

Exemplo 7a: luz com interruptor (v1)

```
DEF LightSwitch TouchSensor { }
DEF LampLight SpotLight { ... }
DEF ToogleButton Script {
  field      SFBool on TRUE
  eventIn    SFBool set_active
  eventOut   SFBool on_changed
  url "javascript:
    function set_active( set, timestamp ) {
      if(set == FALSE) return;
      if(on == TRUE)    on = FALSE;
      else              on = TRUE;
      on_changed = on;
    }"
}

ROUTE LightSwitch.isActive TO ToogleButton.set_active
ROUTE ToogleButton.on_changed TO LampLight.on
```

E o material que representa a luz?

Não tem aspecto diferente quando a luz está ligada e desligada?

Exemplo 7b: luz com interruptor (v2)

```
DEF LightSwitch TouchSensor { }
DEF LampLight SpotLight { ... }
DEF ToogleButton Script { ... }
DEF LampMaterial Material { ... }

DEF ColorSelector Script {
  field      SFCOLOR onColor  1 1 1 # ON - cor branca
  field      SFCOLOR offColor 0 0 0 # OFF - cor preta
  field      SFBool on TRUE
  eventIn    SFBool set_selection
  eventOut   SFBool color_changed
  url "javascript:
    function set_selection( b, timestamp ) {
      if(b == TRUE)    color_changed = onColor;
      else              color_changed = offColor;
    }"
}

ROUTE LightSwitch.isActive TO ToogleButton.set_active
ROUTE ToogleButton.on_changed TO LampLight.on
ROUTE ToogleButton.on_changed TO ColorSelector.set_selection
ROUTE ColorSelector.color_changed TO LampMaterial.set_selection
```

Define-se um nó Script
que retorna cores diferentes
de acordo com o seu estado
(ON/OFF)

- Funcionamento equivalente ao
do botão

Referências

- Norma VRML97 (VRML2.0)
 - <http://www.web3d.org/x3d/specifications/vrml/ISO-IEC-14772-VRML97/>
- **Introduction to VRML 97 (David R. Nadeau)**
 - http://www.siggraph.org/education/materials/siggraph_courses/S98/18/vrml97/slides/mt0000.htm
- Floppy's VRML 97 Tutorial:
 - <http://web3d.vapourtech.com/tutorials/vrml97/>
- Web 3D Consortium – VRML Archives
 - <http://www.web3d.org/x3d/vrml/index.html>