Wann?

Montag
12:15 – 13:45 Kurs
anschließend
Konsultation / Übung

KW	SW	Tag	#	?
17	4	23.4.	1	
18	5	30.4.	2	
19	6	7.5.	-	Kompaktwoche
20	7	14.5.	3	
21	8	21.5.	4	
22	9	28.5.	-	Kompaktwoche / Pfingstmontag
23	10	4.6.	5	
24	11	11.6.	6	
25	12	18.6.	-	Kompaktwoche
26	13	25.6.	7	
27	14	2.7.	8	

„Wiederholung“

Was ist / sind …

… Lingo ?
… die Filmmethapher ?
… Besetzung, Drehbuch und Bühne ?
… ein Sprite ?
… Tweening ?
… Onion-Skinning ?
… Verhalten ?
… Skripte?
… ein Handler ?
… drei Arten Variablen ?
… Nachrichtenhierarchie ?

Beispiel: Sprites verschieben mit der Maus

Verschiedene Lösungsmöglichkeiten


mousefollow_1.dir


mousefollow_2.dir


mousefollow_3.dir


mousefollow_4.dir

Variante 1	Spriteskript mit `mouseWithin` – Handler
Variante 2	Bildskript mit `exitFrame` – Handler
Variante 3	zwei Spriteskripte, eins dauerhaft, das andere temporär am Sprite mit `the scriptInstanceList`
Variante 4	ein Spriteskript, das über `the actorList` nur dann aktiv wird wenn nötig

→ Scriptinstancelist
Jedes Sprite hat eine Skript-Instanz-Liste.
In diese Liste stehen die an dem Sprite angebrachten Skripte. Sowohl solche die per Drag und Drop am Sprite angebracht werden, als auch mit Lingo angebrachte Verhalten. Alle Skripte, die in der Skript-Instanz-Liste eines Sprites stehen, können auf Nachrichten reagieren, die zum Sprite geschickt werden.

→ Actorlist
Globale Filmeigenschaft. Alle in dieser Liste befindlichen Objekte (Skriptinstanzen) erhalten eine `stepframe` – Nachricht zu Beginn eines neuen Frames.

Beispiel: Animation mit Lingo

→ Rückblick – „klassische“ Drehbuchanimation


guybrush - klassisch animiert.dir

Motivation: dynamische Animationen, die nicht im Drehbuch festgelegt werden

→ on the fly erklären


guybrush - mit lingo animiert 1.dir


guybrush - mit lingo animiert 2.dir

Beispiel: Auffangspiel


Auffangspiel.dir

- Puppet-Sprites:
  Mit `sprite(x).puppet = true` wird dem Drehbuch die Kontrolle über den Spritekanal entzogen und der Sprite ist nur über Lingo-Befehle steuerbar.

```
SP = sprite(1)
SP.puppet = true -- "verpuppen" des Sprites
SP.member = member("roter Punkt") -- Darsteller zuweisen
SP.ink = 1 -- Hintergrund transparent
SP.loc = point(50, 100) - Position setzen
```

- Verhaltensskripte können auch mit Lingo an Sprites angebracht werden (statt sie mit der Maus auf ein Sprite im Drehbuch zu ziehen). Man erzeugt eine neue Instanz des Verhaltensskriptes, und fügt diese Instanz zur Skriptinstanzliste des Sprites hinzu.

```
oVerhaltensInstanz = script("Bewegung").new(SP)
SP.scriptInstanceList.add(oVerhaltensInstanz)
```

- Nicht nur eins, sondern (fast) beliebig viele Sprites werden erzeugt, wenn man die „Sprite-Verpuppung“ in einer repeat-Schleife mit mehreren Sprites durchführt. Diese Schleife kann in einen Handler in einem Filmskript eingebettet werden. Das ergibt eine globale Funktion, die mit Lingo-Anweisungen eine gewisse Anzahl Sprites erzeugt:

```
on ErzeugeSprites aiAnzahl

  repeat with i = 1 to aiAnzahl
    SP = sprite(i)
    SP.puppet = true
    SP.member = ... -- usw.
    ...
  end repeat

end
```

- An jedes der auf diese Weise erzeugten Sprites wird genau eine Instanz des Verhaltensskriptes „Bewegung“ angebracht. Jede Skriptinstanz steuert dann das Sprite, an dem es angebracht ist.
  Damit man von einem Skript eine Instanz mit Lingo erzeugen kann, muss das Skript einen speziellen Handler besitzen, den new-Handler. Von diesem Handler muss die Referenz auf die Skripinstanz (me) zurückgegeben werden:

```
on new me
  -- Verschiedene Initalisierungsanweisungen
  return me
end
```

- Handler, die in Verhalten stehen, die an Sprites angebracht sind, können auf verschiedene Arten aufgerufen werden:

  über das Sprite
  ```
  sprite(2).StarteBewegung()
  ```

  über die Skriptinstanz selbst
  ```
  oVerhaltensInstanz.StarteBewegung()
  ```

  durch den sendsprite-Befehl
  ```
  sendsprite( 2, #StarteBewegung )
  ```

  Diese unterschieden sich vor allem in der Fehlerbehandlung.
  Wenn beispielsweise das Sprite die Nachricht doch nicht behandlen kann, weil an ihm es kein Verhalten angebracht wurde, das über einen passenden Handler verfügt, wird bei der ersten Aufrufvariante ein Fehler erzeugt, bei der dritten hingegen nicht.

- In dem Verhaltensskript „Bewegung“ befinden sich folgende Handler, die für gewisse Aktionen verantwortlich sind:

Handler	Aktion
`new`	Erzeugung einer Skriptinstanz, Initialisierung von Propertyvariablen
`StarteBewegung`	stepFrame-Nachrichten erhalten
`StoppeBewegung`	stepFrame-Nachrichten nicht mehr erhalten
`mouseWithin`	Auffangen des Sprites
`mouseLeave`	wieder-los-lassen des Sprites
`stepFrame`	Position des Sprites berechnen und aktualisieren

- Im stepFrame-Handler wird in jedem Frame die neue Position des Sprites berechnet. Das geschieht, indem zwei Vektoren miteinander addiert werden und der resultierende Vektor auf die aktuelle Spriteposition addiert wird.

  die beiden Vektoren sind

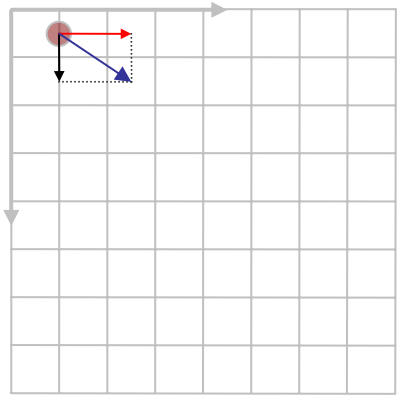
`gpntGravitation`	Gravitation, in einer globalen Variable, Typ `#point`
`ppntGeschwindigkeit`	Geschwindigkeit, in einer Property-Variable, Typ `#point`

AA1-9

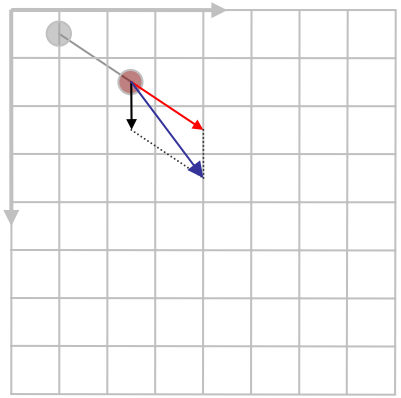
Gravitation
Geschwindigkeit
Gravitation + Geschwindigkeit = neue Geschwindigkeit

```
pntNeueGeschwindigkeit = ppntGeschwindigkeit + gpntGravitation
pSprite.loc = pSprite.loc + pntNeueGeschwindigkeit
ppntGeschwindigkeit = pntNeueGeschwindigkeit
```

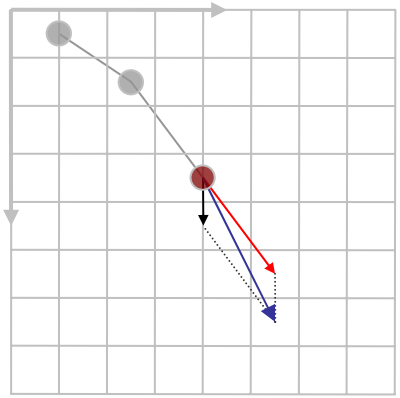
Frame 1



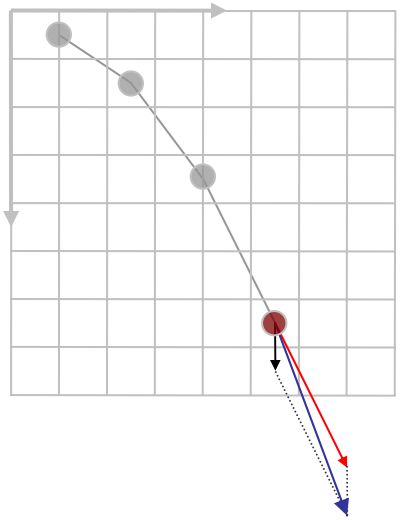
Frame 2



Frame 3

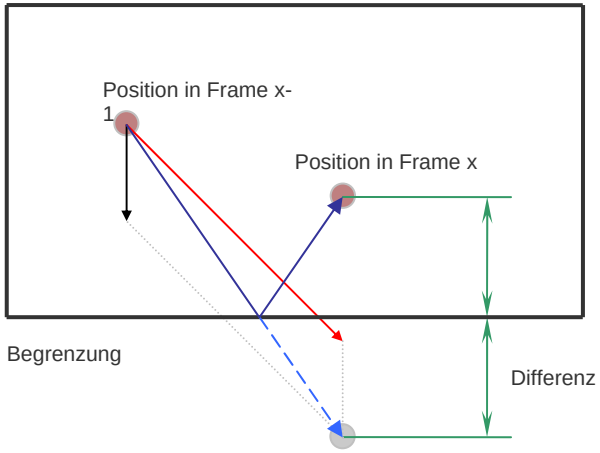


Frame 4



- Nach der Positionsberechnung findet die Reflektionsberechnung statt.

  Dazu wird in jedem neuen Frame für alle 4 Seiten des Begrenzungsrechtecks überprüft, ob sich das Sprite innerhalb oder außerhalb des Rechtecks befindet. Falls es sich außerhalb befindet, hätte es auf dem Weg zu seiner aktuellen Position eigentlich schon reflektiert werden müssen.
  Da hier im Beispiel nur an achsenparallelen Geraden reflektiert wird, ist die Berechnung der reflektierten Position aber einfach über die Differenz aus Begrenzung und Position bezüglich der betrachteten Achse möglich.




```
if pSprite.locV < prBegrenzung.top then
  ppntGeschwindigkeit.locV = -1 * ppntGeschwindigkeit.locV
  diff = prBegrenzung.top - pSprite.locV
  pSprite.locV = prBegrenzung.top + diff
end if

if pSprite.locV > prBegrenzung.bottom then
  ppntGeschwindigkeit.locV = -1 * ppntGeschwindigkeit.locV
  diff = pSprite.locV - prBegrenzung.bottom
  pSprite.locV = prBegrenzung.bottom - diff
end if

-- wenn man diff gleich einsetzt passt die Berechnung auf eine Zeile
if pSprite.locH > prBegrenzung.right then
  ppntGeschwindigkeit.locH = -1 * ppntGeschwindigkeit.locH
  pSprite.locH = 2 * prBegrenzung.right - pSprite.locH
end if

if pSprite.locH < prBegrenzung.left then
  ppntGeschwindigkeit.locH = -1 * ppntGeschwindigkeit.locH
  pSprite.locH = 2 * prBegrenzung.left - pSprite.locH
end if
```

Eventverarbeitung und Nachrichtenhierarchie

Event = Ereignis

Director „selbst“ bzw. der Benutzer (durch Interaktion) generiert „Events“,
daraufhin werden Nachrichten erzeugt und verschickt.

An Wen werden Nachrichten verschickt?
→ An die Skripte, die im Film verfügbar sind und die Nachricht behandeln können.

Wer behandelt eine Nachricht und reagiert somit auf ein Event?
→ Ein für die Nachricht passender *Event-Handler.*

→ Eventhierarchie legt fest, in welcher Reihenfolge in den Skripten nach einem passenden Event-Handler gesucht wird.

Nachrichtenhierarchie kurz gesagt:

→ Primäre Ereignisprozedur, Spriteskript, Darstellerskript, Bildskript und Filmskript.

Mit primären Ereignisprozeduren (Primary Event Handler) kann man durch die linke Maustaste oder die Tastatur generierte Events abfangen und von einem Handler in einem Filmskript behandlen lassen. Primäre Ereignisprozeduren sind:

```
the mouseUpScript, the mouseDownScript,
the keyUpScript, the keyDownScript, the timeOutScript
```

Spriteskripte sind Verhalten, die auf Sprites angewendet werden

Darstellerkripte sind direkt mit einem Darsteller verknüpft (und veraltet)

Bildskripte sind Verhalten, die auf Frames angewendet werden (im Skriptkanal)

Filmskripte liegen nur in den Besetzungen rum und gelten für den gesamten Film

Nachrichtenhierarchie DirectorEvents.dir

Nachrichtenhierarchie UserEvents.dir

Nachrichtenhierarchie (ausführlich)