AS3-Tutorium: Flash: Butterfly 08 character

Wechseln zu:Navigation, Suche

Dieser Artikel ist veraltet und wird künftig evtl. entfernt.

Dieser Artikel erfüllt die GlossarWiki-Qualitätsanforderungen **nur teilweise**:

Korrektheit: 4 (großteils überprüft)	**Umfang**: 3 (einige wichtige Fakten fehlen)	**Quellenangaben**: 5 (vollständig vorhanden)	**Quellenarten**: 5 (ausgezeichnet)	**Konformität**: 4 (sehr gut)

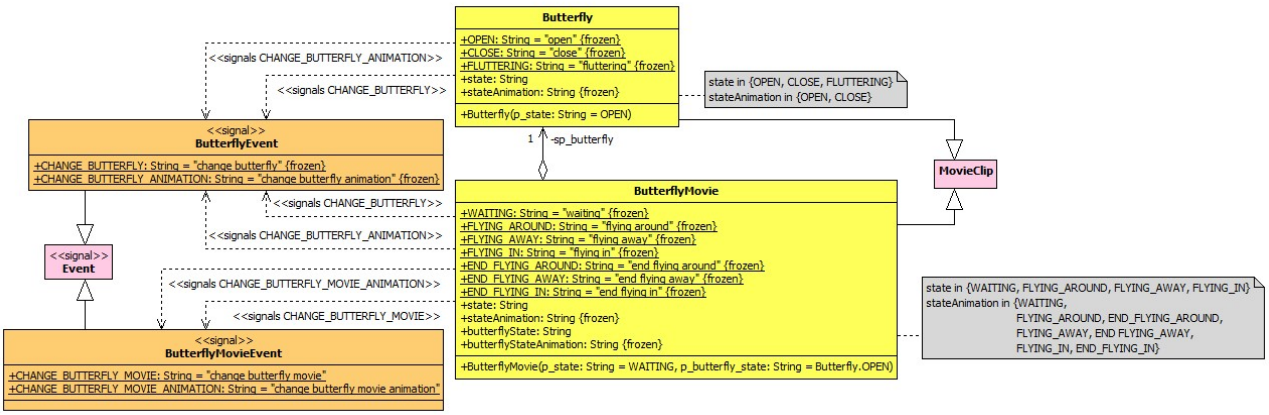
Inhaltsverzeichnis

1 Eine zweite Spielfigur

Nachdem im siebten Teil des Tutoriums das Schmetterlingssymbol in eine Spielfigur umgewandelt wurde, wird nun auch das Symbol `ButterflyMovie` als Klasse von Viewobjekten mit wohldefinierter Schmittstelle implementiert. Dabei wird insbesondere die Progammlogik (fliegen einer via `roundsToFly` vorgegebenen Anzahl von Runden) aus der Klasse `ButterflyMovie` in die Hauptklasse `Main` verlagert.

1.1 Das Datenmodell

Folgendes Klassendiagramm wird in diesem Teil des Tutorium realisiert:



Ordner mit StarUML-Diagramm und von StarUML erzeugtem AS3-Code-Rahmen (SVN-Repository)

Es gibt zwei Klassen zur Visualisierung einer Spielfigur (so genannte View-Klassen):

`Butterfly`, eine Spielfigur, die sich nicht selbstständig vom Platz bewegt, aber verschiedene Zustände annehmen kann.
`ButterflyMovie`, eine animierte Spielfigur, die sich auf vordefinierten Bahnen bewegt und auch verschiedene Zustände annehmen kann.

Beide Klassen sind, da sie Symbolen zugeordnet werden, Subklassen der ActionScript-Klasse `MovieClip`.

Die Attribute und die Signale der Klasse `Butterfly` wurden bereits in Teil 7 des Tutorium ausführlich beschrieben.

Ein Schmetterlingsmovie kann vier Zustände annehmen:

wartend (`WAITING`)
herumfliegend (`FLYING_AROUND`)
wegfliegend (`FLYING_AWAY`)
von außerhalb zum Warteplatz fliegend (`FLYING_IN`)

Dieser Zustand kann über das Attribut `state` gelesen und modifiziert werden. Mit Hilfe des Attributs `stateAnimation` kann erfragt werden, in welchem Zustand sich die Animation gerade befindet. Dieses Attribut ändert sich jeweils zum Ende einer Flugsequenz. Das heißt, das Attribut kann insgesamt sieben Werte annehmen:

wartend (`WAITING`)
herumfliegen wurde beendet (`END_FLYING_AROUND`)
wegfliegen wurde beendet (`END_FLYING_AWAY`)
zum Warteplatz fliegen wurde beendet (`END_FLYING_IN`)

Der Schmetterlingsmovie informiert ebenfalls mittels Signalen interessierte Beobachter sowohl dann, wenn sich der Zustand (`state`) ändert, als auch dann, wenn sich der Zustand der Animation (`stateAnimation`) ändert, d.h., wenn eine Flugsequenz beendet wird. Hierfür wird die Klasse `ButterflyMovieEvent` verwendet.

Auf das Schmetterlingsobjekt, dass dem Schemtterlingsmovie zugeordnet ist, kann von außen nicht zugegriffen werden: Die Beziehung wurde als `private` definiert. Wäre ein Direktzugriff möglich, so könnte man z.B. das Flügelschlagen während des Rundfluges deaktivieren. Außerdem würde der Direktzugriff Verletzungen des Gesetzes von Demeter zulassen.

Als Ersatz für den fehlenden Direkzugriff werden zwei weitere Attribute angeboten:

`butterflyState`: definiert den Fügelzustand für den wartenden Schmetterling (offen, geschlossen, flatternd); wenn er fliegt, flatter er immer
`butterflyStateAnimation`: gibt den aktuellen Flügelzustand preis: offen oder geschlossen

Da auf das Schmetterlingsobjekt nicht direkt zugegriffen werden kann, signalisiert der Schmetterlingsmovie auch die Zustandsänderungen des Schmetterlingsobjektes.

Hier sieht man ein Problem, das sich bei der Beachtung des Gesetzes von Demeter ergibt, sehr deutlich. Ein Objekt, das keinen Direktzugriff auf ein ihm zugeordnetes Objekt ermöglicht, muss als Adapter (Wrapper) für dieses Objekt implementiert werden. Dies hat zur Folge, dass zusätzlicher Code geschrieben werden muss, der im Wesentlichen nichts weiter macht, als Nachrichten weiterzuleiten.

Hier hat diese Vorgehen allerdings einen entscheidenden Vorteil: Es ist dem Benutzer der Klasse `ButterflyMovie` nicht möglich, das Verhalten des Schmetterling in einem unpassenden Augenblick (d.h. während des Fluges) zu verändern.

Im Allgemeinen ist Code, der hilft, Programmierfehler zu vermeiden, Code vorzuziehen, der aufgrund des geringeren Overheads etwas effizienter ist.

1.2 Der Schmetterlings-Movie als Spielfigur

> TO BE DONE

2 Probleme der Implementierung

> TO BE DONE

3 Quellen

Kowarschick, W.: Multimedia-Programmierung
Musterlösung (Flash CS5)
Musterlösung (Flash CS4)
Erweiterte Musterlösung (Flash CS4)
Erweiterte Musterlösung (Flash CS5)

4 SVN-Repository-Verweise

Musterlösung (Flash CS5)
Musterlösung (Flash CS4)
Erweiterte Musterlösung (Flash CS4)
Erweiterte Musterlösung (Flash CS5)

> Dieser Artikel ist GlossarWiki-konform.

Kategorien:
AS3-Tutorium: Flash: Butterfly
Flash-Beispiel
Kapitel:Multimedia-Programmierung:Beispiele