Hausaufgabe 2 – Multimedia-Programmierung

Es soll in Flash ein „Game of Life“ Spiel erstellt werden. Siehe z.B.:

- Beschreibung auf http://www.math.com/students/wonders/life/life.html
- Beschreibung auf http://en.wikipedia.org/wiki/Conway's_Game_of_Life
- Applet auf http://www.ibiblio.org/lifepatterns/

Regeln:

Das Spielfeld ist in quadratische Zellen eingeteilt. Jede Zelle kann entweder „tot“ oder „lebendig“ sein (leer oder ausgefüllt). In jeder Runde verändert sich der Zustand der Zellen gemäß folgender Regeln:

- Eine Zelle mit weniger als 2 Nachbarn stirbt (Einsamkeit)
- Eine Zelle mit mehr als 3 Nachbarn stirbt (Überpopulation)
- Eine tote Zelle mit 3 Nachbarn wird lebendig
- Eine lebendige Zelle mit 2 oder 3 Nachbarn bleibt am Leben

Als Nachbarn gelten die 8 Zellen, die senkrecht, waagerecht oder diagonal angrenzen. Es ist zu beachten, dass als Berechnungsgrundlage für alle Zellen der Zustand der letzten Runde gilt, d. h. in jeder Runde werden zunächst die neuen Zustände aller Zellen berechnet und erst dann werden etwaige Zustandsänderungen ausgeführt.

Anforderungen:

- Das Spielfeld und die einzelnen Zellen werden beides jeweils durch einen MovieClip realisiert, der mit einer ActionScript2-Klasse verknüpft ist.
- Das Spielfeld wird beim Start der Anwendung mittels ActionScript-Code mit den Zellen gefüllt (dynamische Erstellung von MovieClip-Instanzen). Dabei wird dynamisch die Anzahl benötigter Zellen berechnet.
- Die Bühne enthält (neben dem Spielfeld) mehrere Buttons, um das Spiel zu starten (d.h. regelmäßige automatische Ausführung eines Schritts), anzuhalten, einen Einzelschritt auszuführen oder das Spielfeld zu löschen (d.h. alle Zellen in den Zustand „tot“ zu setzen).
- Jede Zelle kann mit der Maus angeklickt werden, um den Zustand der Zelle zu verändern (Wechsel zwischen „tot“ und „lebendig“).
- Geben Sie der Bühne eine individuelle graphische Gestaltung (Farben, Schriftzüge, Hintergrundbild oder ähnliches – braucht nicht aufwändig zu sein).

Lösungsvorschlag:

Wichtig ist, dass für die Ermittlung der Zustände eines nächsten Schrittes die Anzahl der Nachbarn im aktuellen Schritt herangezogen werden muss. Dazu kann in zwei Stufen vorgegangen werden: zunächst werden für alle Zellen die Anzahl ihrer Nachbarn ermittelt und gespeichert (z.B. in einem Attribut `neighbours`). Anschließend werden für alle Zellen die Zustände geändert gemäß der ermittelten Anzahl ihrer Nachbarn. Zur effizienteren Ermittlung der Nachbarn ist es zweckmäßig, die Zellen einmal zu durchlaufen und für jede gefundene lebende Zelle bei ihren 8 Nachbarn die Anzahl der Nachbarn zu erhöhen (anstatt für jede Zelle einzeln alle 8 Nachbarn abzufragen).

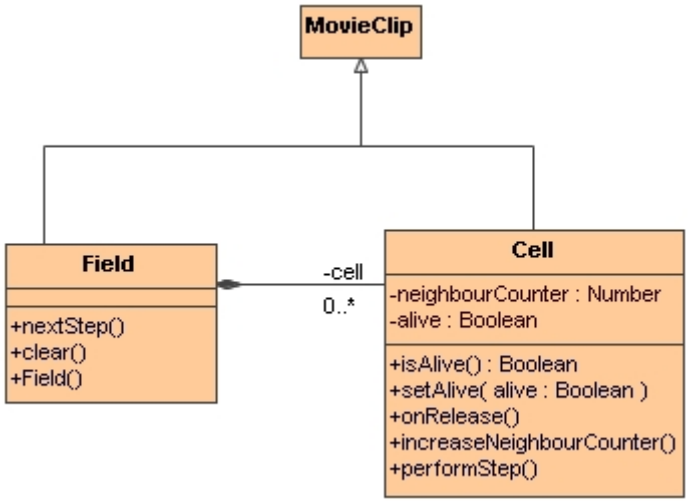

Abbildung 1: AS2-Klassen

Entsprechend ergibt sich folgender Aufbau:
Klasse `Cell`: repräsentiert eine einzelne Zelle auf der Bühne.

- `alive:Boolean`: enthält aktuellen Zustand (tot oder lebendig).
- `neighbourCounter:Number`: wird hochgezählt um Anzahl der Nachbarn zu ermitteln.
- `isAlive()`: liefert aktuellen Zustand zurück.
- `setAlive(alive:Boolean)`: setzt aktuellen Zustand. Dabei muss auch die graphische Repräsentation (zugehöriger MovieClip auf der Bühne) angepasst werden. Am besten einfach im MovieClip zwei Schlüsselbilder erstellen – eines für den Zustand „tot“ (leeres Rechteck) und eines für den Zustand „lebendig“ (ausgefülltes Rechteck). Wenn der Zustand sich ändert, muss zum entsprechenden Schlüsselbild gesprungen werden. Dazu am besten dem Schlüsselbild einen Namen (Label) zuordnen.
- `onRelease()`: Ereignisbehandlungsmethode für Mausklicks. Bewirkt eine Änderung des aktuellen Zustands durch Aufruf von `setAlive()`.
- `increaseNeighbourCounter()`: wird aufgerufen, um `neighbourCounter` um 1 zu erhöhen
- `performStep()`: berechnet den neuen Zustand basierend auf `neighbourCounter` und setzt ihn mittels `setAlive()`.

Klasse Field: repräsentiert das Spielfeld, das die einzelnen Zellen enthält.

- `cell`: enthält die einzelnen Zellen (d.h. Instanzen von Cell). Am besten als zweidimensionales Array (in Flash: Array von Arrays) realisieren.
- `Field()`: Konstruktor; erzeugt Instanzen der Zelle, platziert diese auf der Bühne und speichert sie im Attribut `cell`. Die benötigte Anzahl von Zellen wird dabei berechnet aus der Höhe bzw. Breite (Attribute `_width` und `_height`) des Spielfelds auf der Bühne und der Höhe bzw. Breite einer einzelnen Zelle (jeweils aufrunden).
- `nextStep()`:durchläuft alle Zellen. Für jede gefundene lebende Zelle wird bei allen 8 Nachbarn `increaseNeighbourCounter()` aufgerufen. Anschließend werden nochmals alle Zellen durchlaufen und jeweils `performStep()` aufgerufen.
- `clear()`: durchläuft alle Zellen und setzt ihren Zustand auf „tot“.

FLA-Hauptdatei:
Auf der Bühne befinden sich das noch leere Spielfeld (Instanznamen nicht vergessen) und die Schaltflächen. In der Bibliothek befinden sich ein MovieClip für die Zellen und einer für das Spielfeld; jeweils mit den zugehörigen Klassen verknüpft.
Die Ereignisbehandlungsmethoden für die Schaltflächen auf der Bühne (Komponente `Button`) werden einfach im ersten Frame definiert.

- Button „Step“: führt einen einzelnen Schritt aus, d.h. Aufruf der Methode `nextStep()` von `Field`.
- Button „Run“: führt automatisch (z.B. alle 10 ms) `nextStep()` aus. Dazu kann die eingebaute ActionScript-Funktion `setInterval()` verwendet werden.
- Button „Stop“: hält die automatische Ausführung an mittels der eingebauten ActionScript-Funktion `clearInterval()`.
- Button „Clear“: löscht das Spielfeld durch Aufruf von `clear()`.

Abgabe:

- Einzureichen sind FLA- und SWF-Datei.
- Falls Sie (zu Hause) Flash 8 verwenden, speichern Sie die Datei kompatibel zu Flash 7 (*Speichern unter…*)!
- Benennen Sie Ihre Dateien mit ***Vorname.Nachname***!
- Es werden nur individuelle Lösungen angenommen!
- Schicken Sie Ihre Lösung bis spätestens **17. Mai 2007, 23:59 Uhr** per Email an Andreas Pleuß.