OSTBAYERISCHE TECHNISCHE HOCHSCHULE REGENSBURG

ReMIC Regensburg Medical Image Computing

Prof. Dr. Christoph Palm

Computergraphik Sommersemester 2014

Projektarbeit

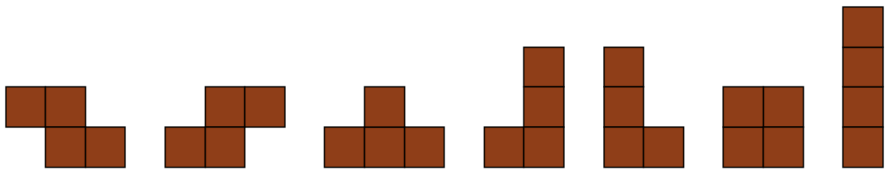
Im Rahmen der Lehrveranstaltung gibt es neben den Übungsblättern eine Projektarbeit. Die Arbeit soll in Gruppen (jeweils 2-3 Teilnehmer) bearbeitet werden.
Damit Sie sich bei der Umsetzung möglichst frei bewegen können, gibt es keine strikten Vorgaben.

Aufgabenstellung

Programmieren Sie Tetris unter Verwendung des *THREE.js*-Frameworks. Sorgen Sie dafür, dass alle bekannten Mechanismen richtig funktionieren.

Elemente

- Steine
  Die sieben Steine setzen sich jeweils aus vier Blöcken zusammen:



- Spielfeld
  Das klassische Spielfeld besteht aus einem 10x20 Gitter.
- GUI
  In einer GUI wird der Punktestand, der aktuelle Schwierigkeitsgrad (Level 1 bis 9) und der nächste Stein angezeigt.

Spielmechanismen

- Steuerung

Der aktuelle Stein kann mit den Pfeiltasten nach links, rechts und unten bewegt sowie im Uhrzeigersinn rotiert werden.

- Kollision

  Kollidiert ein Stein mit dem unteren Spielfeldrand oder mit bereits auf dem Spielfeld befindlichen Steinen, bleibt der aktuelle Stein stehen und der nächst Spielstein erscheint am Feld. Auch Kollisionen in horizontaler Richtung sollen berücksichtigt werden und eine Bewegung des Spielsteins in die entsprechende Richtung verhindern. Nicht nur Bewegungen, sondern auch Rotationen können eine Kollision verursachen.

- Bewegung

  Nach einem festgelegten Zeitintervall, welches vom Schwierigkeitsgrad abhängt, bewegt sich ein Spielstein um eine Einheit nach unten.

- Abbauen von Linien

  Ist eine Zeile komplett mit Blöcken gefüllt, wird die Zeile ausgelöscht und alle darüber liegenden Blöcke entsprechend nach unten bewegt.

- Punktestand und Schwierigkeitsgrad

  Der Punktestand wird mit jedem Zeilenabbau um 100 erhöht. Ausserdem wird der Schwierigkeitsgrad, der die Fallgeschwindigkeit des aktuellen Steins erhöht ebenfalls durch den aktuellen Punktestand bestimmt.

- Spielende

  Das Spielende ist erreicht, sobald ein Block den oberen Spielfeldrand überschreitet.

Hinweise

- Input

  Verwenden Sie für die Verarbeitung von Tastatureingaben folgendes

Funktionsgerüst:

```
document.onkeydown = function(evt)
{
    evt = evt || window.event;
    switch (evt.keyCode)
    {
        case 37: /* links-taste behandeln */ break;
        case 38: /* oben-taste behandeln */ break;
        case 39: /* rechts-taste behandeln */ break;
        case 40: /* unten-taste behandeln */ break;
    }
}
```

- Klassen

  JavaScript erlaubt die Verwendung des Klassenkonzepts. Anstatt eines eigenen Schlüsselwortes wird eine Klasse über eine Funktion definiert, die wiederum Attribute und Funktionen enthalten kann. Hier wird nochmals das hohe Level der Sprachdynamik deutlich. Das nachfolgende Codesnippet skizziert, wie Klassen in JavaScript definiert werden:

```
var Field = function()
{
  var width;
  var height;
  this.build = function(w,h)
  {
    width = w;
    height = h;
    ...
  }
  ...
```

  Die Klasse lässt sich dann beispielsweise in der init-Funktion folgendermaßen instanziieren und ansprechen:

```
function init()
{
  ...
```

```
    field = new Field();
    field.build(10,20);
    ...
}
```

- Namespaces/Packages

  Auch Namespaces bzw. Packages kann man in Javascript umsetzen. Betrachten Sie dazu folgendes Codebeispiel:

```
var namespace = {};

namespace.attribute = "beliebiges Attribut";
namespace.subnamespace = {};
namespace.subnamespace.attribute = 42;
```

  Eine Variable wird als Container interpretiert, sobald als Wert das Klammerpaar zugewiesen wird. Ausserdem können solche Container beliebig tief verschachtelt werden.
  Beachten Sie, dass Containerelemente nicht mit dem Schlüsselwort *var* ausgezeichnet werden. Solche Container erlauben es, globale Variablen sinnvoll zusammenzufassen:

```
var tetris = {};
tetris.score = 0;
tetris.level = 1;
tetris.field = 0;
tetris.currentblock = 0;
tetris.nextblock = 0;
...
```

- Struktur

  Da die Logik hinter Tetris relativ umfangreich ist, ist eine durchdachte Strukturierung sehr wichtig. Es bietet sich beispielsweise an, eigene Klassen für Spielsteine und das Spielfeld zu verwenden. Ordnen Sie auch Funktionen sinnvoll den Klassen zu. Überlegen Sie, welche Ereignisse im Spielverlauf eintreffen können und reagieren Sie in entsprechenden Methoden darauf. Wenn Sie viele kleine Funktionen verwenden, gestalten Sie Ihren Code übersichtlicher und verständlicher.

- Implementierung

  Sie können eine beliebige Übung als Grundlage verwenden. Lediglich bei der *update*-Funktion muss eine zeitliche Verzögerung eingebaut werden, sodass Spielsteine abhängig vom aktuellen Schwierigkeitsgrad erst nach Ablauf eines Zeitintervalls um eine Einheit nach unten bewegt werden. Das können Sie zum Beispiel durch Verwendung einer Variablen *frame_number* umsetzen. Bei jedem Aufruf der *update*-Funktion wird die Variable *frame_number* inkrementiert und erst wenn ein gewisser Wert erreicht ist, also eine festgelegte Anzahl an Frames verstrichen ist, soll der Spielstein automatisch nach unten bewegt werden.

  Was die grafische Komponente angeht, gibt es keine strikten Vorgaben. Sie können die Spielsteine mithilfe von 2D Linienobjekten oder aus 3D Objekten zusammensetzen. Die Schwierigkeit in diesem Projekt besteht in erster Linie in der Anwendungslogik und dem dahinterliegenden Datenmodell. Haben Sie eine solide Basis vorliegen, können Sie die grafische Repräsentation nachträglich beliebig anpassen.

  Um das Spielfeld zu verwalten, bietet sich die Verwendung eines zweidimensionalen Arrays an. So können Sie den Zustand des Spielfelds für jede Position unterbringen. Betrachten Sie dazu folgenden Codeausschnitt:

```
// eindimensionale arrays:
var  a = new Array();
a.push(1);
a.push(2);
a[0] = 3;

// zweidimensionale arrays:
var b = new Array();
b.push(new Array(2));
b.push(new Array(2));

b[0][0] = "eins";
b[0][1] = "zwei";
b[1][0] = "drei";
b[1][1] = "vier";
```

Checkliste

Überprüfen Sie folgende Mechanismen auf richtige Funktionsweise:

- Bewegung ausserhalb des Spielfelds verhindern
- Kollision mit anderen Spielsteinen
- Kollision nach Rotation (Spielfeldrand und andere Blöcke)
- Eliminieren mehrerer Zeilen
- Punktestand nach Zeilenabbau

Bewertung

Das Basiskriterium hinsichtlich der Bewertung ist die Umsetzung der Spielmechanismen. Um jedoch die bestmögliche Bewertung zu erreichen, müssen Sie Ihrem Projekt eine eigene Note verleihen. Die Möglichkeiten dafür sind quasi unbegrenzt. Sie können den Objekten Schatten hinzufügen, die Spielsteine texturieren eine Hintergrundszene aufbauen oder andere visuelle Effekte verwenden. Solche Features sind durch das *THREE.js*-Framework einfach zugänglich, erfordern aber ein gewisses Maß an Recherche. Nützlich ist die Dokumentation des Frameworks sowie ein Blick in die zahlreichen Samples, die im Ordner *examples* des *THREE*-Ordners zu finden sind. Seien Sie kreativ und schmücken Sie Ihr Projekt mit ausgesuchten Features aus um die bestmögliche Bewertung zu erhalten!
Hinweis: Da es sich um einen Kurs in Computergraphik und nicht in Musik handelt, sind akustische Effekte erlaubt, werden aber nicht mitbewertet.

Folgende Teilaufgaben werden einzeln bewertet:

1. Spielfeld und GUI mit Punktestand und Levelanzeige wird dargestellt
2. sieben Steine fallen in zufälliger Reihenfolge in das Spielfeld (Bewegung nach unten)
3. aktueller Stein kann links / rechts bewegt und rotiert werden
4. Kollision mit Spielfeldrand oder unteren Steinen wird erkannt und darauf entsprechend reagiert

5. Kollision links/rechts wird erkannt und darauf entsprechend reagiert
6. Abbau einer vollständigen Zeile, auch mehrere Zeilen
7. Erkennung des Spielendes, falls oberste Linie überschritten

Wir gehen von der Basisnote: 4,3 aus. Für jede erfüllte Teilaufgabe gibt es -1 Teilnote (z.B. von 4,3 auf 4,0 oder von 1,7 auf 1,3). Für die eigene Note gibt es je nach Schwierigkeitsgrad bis zu -3 Teilnoten. Sind alle Teilaufgaben erfüllt wird eine 2,0 erreicht. Mit einer kreativen eigenen Note dann eine 1,0.

Bitte beachten Sie, dass individuelle Noten vergeben werden.