Institut für Informatik
Prof. Dr. rer. nat. Oliver Vornberger
Nico Marniok, B.Sc.
Erik Wittkorn, B.Sc.

Universität Osnabrück, 23.04.2013
http://www-lehre.inf.uos.de/~ai3de
Abgabe bis 03.05.2013, 23:59 Uhr

Übungen zum Aufbau interaktiver 3D-Engines

Sommersemester 2013

Blatt 2

Übungsbetrieb

In der Übung am Dienstag wird ein Aufgabenblatt verteilt, das bis zum Freitag um 23:59 Uhr der darauffolgenden Woche zu bearbeiten ist. Die Aufgabenblätter befinden sich auch auf der Veranstaltungswebseite (http://www-lehre.inf.uos.de/~ai3de) sowie im Stud.IP.

Die Übungen finden dienstags von 12:15 – 13:45 in Raum 31/304 statt. In der Übung wird alle zwei Wochen (beginnend am 23.04.2013) eine Kurzpräsentation einer Lösung des letzten Blattes und die Vorstellung des nächsten Übungsblattes durchgeführt. An den restlichen Terminen wird alle zwei Wochen (beginnend am 16.04.2013) ein Austausch der Teilnehmer und Hilfestellung zum laufenden Übungsblatt stattfinden.

Testatbetrieb

Jede Gruppe wird mindestens einmal ihre Lösung wie oben beschrieben während einer Übung vorstellen. Neben der Besprechung des aktuellen Lösungsverlaufes in der Übung wird es für jede Gruppe ein Feedback per Mail geben.

Mailingliste

Jeder Teilnehmer, der sich bis zum 08.04.2013 in der Veranstaltung im Stud.IP eingetragen hat, wurde automatische bei der Mailingliste angemeldet. Die restlichen sollten dies unter http://list.serv.uni-osnabrueck.de/mailman/listinfo/ai3de tun. Über die Mailingliste werden Änderungen im Vorlesungs- oder Übungsbetrieb, sowie eventuelle Fehlerkorrekturen der Aufgabenblätter mitgeteilt. Zudem soll sie als Hilfestellung dienen, damit Probleme untereinander diskutiert werden. Dazu einfach eine Mail an ai3de@list.serv.uni-osnabrueck.de .

Benotung

Die ordentliche Bearbeitung der Arbeitsblätter und Mitarbeit und Anwesenheit bei den Übungsterminen geht zu 1/5 in die Endnote ein.

Quest 1: MagicQuest einrichten (5%)

Folgen Sie diesen Punkten, um das Projekt einzurichten:

1. Führen Sie die gleichen Schritte durch, wie auf Folien 19 und 20 der zweiten Übung beschrieben, mit dem Unterschied, dass Sie statt dem *TestBed* Projekt *MageQuest* aus dem Ordner *aufgabencode/uebung03/* verwenden.
2. Eventuell müssen noch die LWJGL-Libraries eingebunden werden.

Quest 2: eine Baumkrone einfügen (15%)

Folgen Sie diesen Punkten, um eine Baumkrone zu erstellen:

1. Machen Sie sich mit den Dateien im *Levels*-Ordner vertraut
2. Machen Sie sich mit den Dateien *cube** im *Meshes*-Ordner vertraut, sie definieren die verschiedenen Texturierungen eines Würfels.
3. Machen Sie sich mit der Datei *CubeActor.xml* im *Actors*-Ordner vertraut, sie dient als Referenz für einen *Cube-Actor*.
4. Machen Sie sich mit der `loadLevel`-Methode in der *MageQuestLogic.java* vertraut und wie in der Level-Klasse aus einer <Level> - Node ein Level-Objekt geparst wird
5. Implementieren Sie in der `createActorsFromLevel`-Methode der *MageQuestUtil.java* den Fall, dass der Typ *leaves* auftritt. Und zwar so, dass automatisch mehrere Actors mit *cube_leaves.mtl*-Material und den richtigen Koordinaten erstellt werden, sodass sie eine Baumkrone darstellen.

Wichtig Hinweise

- Um eine generierte XML-Node zu überprüfen bietet sich die Methode `XMLUtil.printDocument(document, (Outputstream)System.out)` an

Quest 3: Movement-Events und Prozesse (60%)

3.1 Neue Events für die Rotation erstellen (30%)

1. Machen Sie sich mit den Events in der *MoveEvents.java* vertraut, wie sie vom *InputListener.java* verarbeitet werden und wie sie in der *magequest_defaultconfig.ini* mit Tasten verknüpft wurden.
2. Erstellen Sie vier neue Events für eine Rotation nach links, rechts, oben, unten (Key-Namen: LEFT, RIGHT, UP, DOWN) und stelle in der Config Tasten für sie ein.
3. Registrieren Sie den *InputListener* für die neuen Events und verarbeiten Sie sie analog zu den Bewegungs-Events.
4. Implementieren Sie in der *PlayerController.java* analog zur Bewegung eine neue Methode, die den yprRotation-Vector entsprechend der Rotationsrichtung verändert wird und die Pose des Spielers aktualisiert. Eine Rotation enstpricht einer 90°-Drehung (entspricht PI/2 im Bogenmaß, was wir benutzen).
5. Um bestimmte Teile von PI für die Rotation zu bekommen, bietet sich die Klasse *MathUtil* mit `MathUtil.PI_*` an.

3.2 Bewegungsrichtung anpassen (10%)

1. Manipulieren Sie die *Vector*-Objekte moveFrontBack und moveRightLeft bei einer Rotation so, dass auch nach einer Rotation die Bewegungs-Tasten WASD wie gewohnt funktionieren.

3.3 Kollisionstest (5%)

1. Implementieren Sie die Methode `movePlayer` in der *MageQuestLogic.java* so, dass der Spieler nur bewegt wird, wenn sich im Level an der Zielposition kein Objekt außer einem Pickup-Objekt befindet und unter der Zielposition kein Wasser ist. Außerdem darf sich der Spieler nur innerhalb der Levelbegrenzungen bewegen. Benutzen Sie dazu Informationen aus dem Level-Objekt.

3.4 Flüssige Rotation (15%)

1. Der Spieler soll sich nicht nur flüssig fortbewegen können, sondern auch flüssig rotieren.
2. Implementieren sie im *PlayerController*-Prozess analog zur flüssigen Fortbewegung die Interpolation einer Rotation.
3. Bestimmen Sie dazu erst den Ziel-Rotationsvektor und die Differenz zur aktuellen Rotation. Rotieren Sie dann den Spieler in der `onUpdate`-Methode stückweise zur Zielausrichtung.
4. Die Bewegungsrichtung soll wie in 3.2 nach der Rotation richtig angepasst sein.
5. Bevor eine neue interpolierte Bewegung oder Rotation durchgeführt werden kann, muss der vorherige Vorgang abgeschlossen sein.

Wichtige Hinweise:

- Man vergisst leicht, einen EventListener wie den InputListener für neue Events zu registrieren.
- Eine neue eindeutige ID für ein Event kann man unter http://www.guidgenerator.com/ generieren lassen.
- Ein gesamter Konfigurationsblock der config.ini (z.B. [CONTROLS]) wird von der defaultconfig überschrieben, deshalb Einstellungen immer auch in der defaultconfig einfügen.

Quest 4: Actor-Components, Events und Prozesse (35%)

1. Machen Sie sich mit der Klasse *AttributeContainerComponent.java* vertraut und wie sie für den Spieler in der *MageQuestScene1.xml* für den Spieler definiert wird.
2. Ähnlich wie bei der Baumkrone implementieren Sie die Hilfsmethode `createPickupActor(x, y, z)` der *MageQuestUtil.java* den Fall des types "pickup" so, dass an der Position dieses Level-Elements ein Actor mit der Actor-Referenz *SphereActor.xml* ensteht.
    a. Bei diesem Actor soll jeweils mit einer Wahrscheinlichkeit von 50% eine rote, oder blaue Kugel entstehen.
    b. Für diesen Actor soll eine *AttributeContainerComponent*-Komponente definiert werden, die bei einer blauen Kugel 5 Mana und bei einer roten 5 Health enthält.
    c. Für diesen Actor soll eine leere Pickup-Komponente definiert werden.
3. In der Methode movePlayer in der MageQuestLogic.java befindet sich schon der Ablauf, dass ein CollectPickupEvent geworfen wird, wenn der Spieler sich zu der Position eines Pickups bewegt. Es startet einen *PickupProcess*. Überprüfen Sie die Lauffähigkeit dieses Prozesses, indem Sie bei jeder Veränderung der Spieler-Attribute den aktuellen Stand von Mana und Health ausgeben.
4. Das Pickup soll verschwinden, wenn es benutzt wurde.
    a. Erstellen Sie in der LogicEvents.java ein *DeleteLevelActorEvent*. Es soll die Klasse ActorEvent erweitern.
    b. Wenn der Pickup-Prozess die Attribute des Spielers vollständig aktualisiert hat, soll dieses Event mit der ID des Pickup-Actors geworfen werden.
    c. Die MageQuestLogic soll auf dieses Event hören und mit `GameApp.getGameLogic.deleteActor(int actorID)` die visuelle Repräsentation des Pickups entfernen und die IDs im Level-Objekt aktualisieren.

SideQuest: (15%)

1. Erstellen Sie einen *RespawnPickupProcess*, dem man im Konstruktor eine Position und eine End-Zeit in Sekunden angegeben werden kann.
2. Der Prozess soll nach dem Aufsammeln eines Pickups mit pickupProc.attachChild(respawnPickup); an den *PickupProcess* gehängt werden.
3. Dieser Prozess wartet die angegebenen Sekunden und wirft dann das neu zu erstellende *CreatePickupEvent*, das die Position als Daten mitbekommt.
4. Wenn dieses Event in der *MageQuestGameLogic* ankommt, wird mit der Hilfsmethode `MageQuestUtil.createPickup(x, y, z)` ein neues Pickup erstellt, das wieder sichtbar und aufsammelbar ist.

Wichtige Hinweise

- Der Actor, der eine bestimmte Komponente besitzt, lässt sich mit `*Component.getOwner()` abrufen.

Viel Erfolg ☺