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

Universität Osnabrück, 07.05.2013
http://www-lehre.inf.uos.de/~ai3de
Abgabe bis 17.05.2013, 23:59 Uhr

Übungen zum Aufbau interaktiver 3D-Engines

Sommersemester 2013

Blatt 3

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: Das neue MageQuest einrichten und TestGUI-Element einfügen (10%)

Folgen Sie diesen Punkten, um das Project einzurichten und ein erstes GUI-Element zu erstellen:

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 *MageQuest2* aus dem Ordner *aufgabencode/uebung05/* verwenden.
2. Vergessen Sie nicht, die LWJGL-Libraries einzubinden und die neuste Version der Engine aus dem Repository zu beschaffen.
3. Vergessen Sie nicht, sich den aktuellen Engine-Code und Resourcen zu beschaffen.
4. Erstellen Sie ein zusätzliches GUI-Element in der Datei *Gui/MageQuestGUI.xml*, z.B. ein schließbares Window mit einem Label-Bild oder einem Label-Text, der die Steuerung des Spiels erklärt.
5. Für die weitere Bearbeitung sollte dieses neue Element unsichtbar gemacht werden, damit es nicht nervt.

Quest 2: Balken für mehr Attribute erstellen (25%)

Folgen Sie diesen Punkten, um der aktuellen Statusanzeige mehrere Attribute hinzuzufügen:

1. Machen Sie sich mit der Klasse *MageQuestHumanView.java* vertraut, hier werden benutzerdefinierte GUI-Komponenten erstellt.
2. Machen Sie sich mit der Klasse *AttributeBars.java* vertraut, sie ist unsere selbst erstellte Statusanzeige.
3. Betrachten Sie, wie die AttributeContainerKomponente des Spielers in der *Levels/MageQuestScene1.xml* über den Namen mit dem *AttributeBars*-Element in der *Gui/MageQuestGUI.xml* verknüpft ist und wie die Farben für die Balken definiert werden.
4. Machen Sie sich damit vertraut, wie in der Initialisierung der AttributeContainerKomponente das *AttributeBars*-Object beschafft und gefüllt wird und in der *addValueToAttribute*-Methode aktualisiert wird.
5. Machen Sie sich mit der Klasse *MageQuestComponentDrawer.java* vertraut und implementieren Sie die Methode *createAttributeContainer* so, dass eine dynamische Statusanzeige von bis zu drei Balken gezeichnet wird, deren Balkenlänge sich an dem Wert des Attributs orientiert. Fügen Sie zum Test der AttributeContainerKomponente des Spielers ein paar Attribute hinzu und betrachten Sie die Auswirkungen.

Quest 3: Map (25%)

Folgen Sie diesen Punkten, um die Karte erscheinen zu lassen und sie zu vervollständigen:

1. Machen Sie sich mit der Klasse *Map.java* vertraut, sie initialisiert die Karte und leitet Events weiter. Das Objekt wir am Ende der *loadLevel(String resource)* Methode erstellt.
2. Machen Sie sich mit der Klasse *MapGUI.java* vertraut, sie stellt die Karte als GUI-Element dar.
3. Legen Sie in der config eine beliebige Taste für das *MapToggleMapEvent* (in der *MapEvents.java* definiert) fest und testen Sie, ob die Karte wirklich erscheint.
4. Implementieren Sie in der Klasse *MapGUI.java* die Mathode *initMapButtons(int yLevels)*. Für jedes Stockwerk (yLevel) soll hier analog zur Methode *createMapPlayerLabel(String resource)* ein neues *MapButton* GUI-Element erstellt werden.
    a. Der *MapButton* wird in der *MageQuestHumanView* nur mit seinem Button-Text erstellt. Erweitern Sie diese Erstellung und die Klasse *MapButton.java* so, dass jedem Button das richtige Stockwerk (yLevel) zugewiesen wird.
    b. Implementieren Sie in der Klasse MapButton.java die Methode *fireAction()* so, dass bei einem Klick beim *MapGUI*-Objekt die Methode *setCurrentYLevel(int currentYLevel)* mit der richtigen Stockwerks-Nummer aufgerufen wird und das *MapGUI*-Element neu gezeichnet wird. Überlegen Sie dazu, wie der *MapButton* mit dem *MapGUI*-Objekt kommunizieren kann.
5. In der *repaint()* Methode des *MapGUI* Objects sollen Sie implementieren, dass das korrekte Label aus der *playerLabels-ArrayList* an der korrekten Position angezeigt wird. Die Auswahl des Labels hängt davon ab, ob sich der Spieler und die Karte im gleichen Stockwerk befinden (256_playerMage.png), die Karte sich in einem höheren Stockwerk befindet (256_playerMage_up.png) oder sich in einem niedrigeren Stockwerk befindet (256_playerMage_down.png).

Mögliche Erweiterungen:

- Eine automatisch scrollende Minimap
- Kombination aus Karten Elementen unter dem Spieler und auf gleicher Höhe

Quest 4: AchievementSystem (20%)

Folgen Sie diesen Punkten, um ein rudimentäres Achivementsystem zu erstellen:

1. Definieren Sie in der GUI-XML ein schließbares Window-Element mit einem Bild, das beim Erreichen des Achievements gezeigt werden soll. Zu Beginn sollte es nicht sichtbar sein.
2. Erstellen Sie eine neue Klasse *AchievementSystem.java*, die folgenden Eigenschaften entspricht:
    a. Beim Initialisieren wird das *Achievement-Window* aus dem *Gui* angefordert und mit einer Klassen-Variable referenziert.
    b. Ein Objekt dieser Klasse hört auf ein beliebiges Event (z.B. *PickupCollectedEvent*).
    c. Beim Eintreffen dieses Events wird eine Zählvariable erhöht und beim Erreichen eines Maximalwerts wird das *Window* mit der Methode *show()* eingeblendet.
3. Ein Objekt dieser Klasse wird am Ende der *loadLevel(String resource)* Methode der MageQuestGameLogic.java erstellt und beim EventManager registriert.

Mögliche Erweiterung:

- Definition von Achievements in einer XML (Bild, Text, Anzahl der Punkte, Event auf das gehört werden soll)
- Gesamtübersicht der gesammelten Achievements mit kleinen Bildern und Gesamtpunktzahl als neues GUI-Element

Quest 5: Der sich neigende Zauberstab (20%)

Folgen Sie diesen Punkten, um den Zauberstab des Magiers zu animieren:

1. Machen Sie sich damit vertraut, wie der Zauberstab in der MageQuestScene1.xml definiert ist und wie der Vaterknoten referenziert wird.
2. Die Pose des Zauberstabs wird schon in der PlayerController-Klasse in der Methode setWeapon(Actor weaponActor) gesetzt.
3. Implementieren Sie ein neues MoveEvent, dass beim Druck der Leertaste (SPACE) geworfen wird und im PlayerController durch ein langsames Neigen des Zauberstabs um 45° vom Spieler weg visualisiert wird.
4. Am Ende dieser Animation soll der Zauberstab wieder in seiner Ausgangslage sein.
5. Überprüfen Sie, ob die Animation durch das normale Rotieren des Spielers gestört wird.

Viel Erfolg ☺