Studienleistung 3 - Lunar Lander

Abgabe

Geben Sie Ihre Lösung bis Montag, den *20.01.20*, *23:55 Uhr* über GRIPS ab.

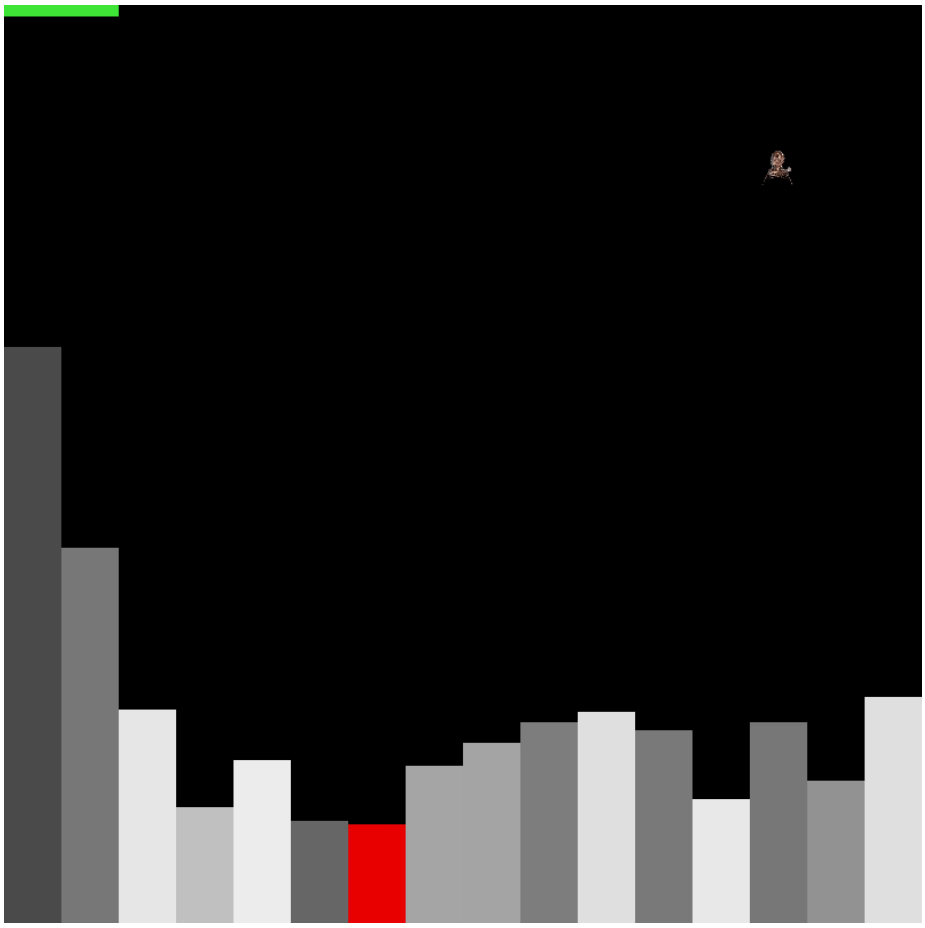
Allgemeine Hinweise zur Studienleistung

In dieser Studienleistung implementieren Sie ein interaktives Spiel. Um die Aufgaben zu bearbeiten, müssen Sie zuerst das Projekt *OOP-Studienleistung-WS19-03-Starterpaket.zip* in IntelliJ öffnen. **Nutzen Sie zum Lösen der einzelnen Aufgaben die bereitgestellten Klassendateien**. Zum Einreichen Ihrer Aufgaben nutzen Sie die entsprechende Funktion in GRIPS. Falls Sie Problemen mit dem Starterpaket oder dem Einreichen der Aufgabe haben, können Sie sich in den Handouts auf GRIPS informieren.

Achtung: Eine Verlängerung der Abgabefrist ist nicht möglich. Einreichungen die uns (zu spät) per E-Mail erreichen, werden nicht mehr berücksichtigt. Alle nicht eingereichten Aufgaben werden mit **nicht bestanden** bewertet. Testen Sie den Upload am besten schon vor Ablauf der Frist in Ruhe: Sie können bis zum Abgabetermin beliebig viele neue Lösungen einreichen.

Bewertungskriterien: Für die gesamte Studienleistung gilt, dass die eingereichten Lösungen nur die in der Aufgabenstellung beschriebenen Probleme lösen sollen. Lassen Sie keinen Teil der jeweiligen Aufgabe weg und interpretieren Sie die Fragestellung nicht selbstständig. Bewertet wird, in wie weit Sie das beschriebene Problem vollständig lösen. Wenn Sie die Aufgaben erfolgreich bearbeitet haben, können Sie Ihre Lösung gerne kreativ gestalten und erweitern; achten Sie dabei darauf, dass die eigentlichen Anforderungen weiterhin erfüllt bleiben.

Die **Qualität Ihres Codes** fließt in die Gesamtnote mit ein: Nutzen Sie Dekomposition um Ihre Programme übersichtlich zu gestalten. Verwenden Sie sinnvolle Bezeichner für Variablen und Methoden und kommentieren Sie ausreichend. Beachten Sie dazu die Kriterien für guten und schlechten Code, die in der Vorlesung erwähnt wurden.



Lunar Lander

In dieser Studienleistung implementieren Sie ein Lunar-Lander-Spiel. SpielerInnen steuern eine Spielfigur, den Lander, der durch die Schwerkraft nach unten gezogen wird, und versuchen diesen zu landen. Eine Schwierigkeit dabei ist, dass der Lander nur begrenzt Treibstoff hat, der durch jeden Lenkimpuls verbraucht wird. Ist der Treibstoff aufgebraucht, bevor der Lander sicher gelandet ist, stürzt er ab. Die Mondoberfläche besteht aus Rechtecken zufälliger Höhe und fester Breite. Eines dieser Rechtecke repräsentiert die Landeplattform.

Klassenmodellierung

Im Starterpaket sind bereits zwei Klassen angelegt, die Sie aus Übungen mit der `GraphicsApp` bereits kennen: - `LunarLander` enthält die `GraphicsApp`-

Anwendung mit `initialize()` und `draw()` - `LunarLanderLauncher` enthält die `main()`-Methode

Ihre Aufgabe ist es, die Spielelemente sinnvoll in weitere Klassen abzubilden. Die Hauptbestandteile der Anwendung sind die **Mondoberfläche** mit **Landeplattform**, der **Lander** und die **Treibstoffanzeige**. In der `LunarLander`-Klasse sollte Code zur Initialisierung, Zeichenaufrufe und Code zur Weitergabe der `KeyPressedEvents` liegen. Der Spielzustand kann hier verwaltet werden.

Vorgaben

Mondoberfläche Zeichnen Sie die Mondoberfläche mit Rechtecken zufälliger Höhe und fester Breite. Damit die Oberfläche nicht ganz so wild aussieht, nutzen Sie die folgende Methode, um die erzeugten Höhenwerte zu glätten:

```
private void smoothPoints(ArrayList<Integer> points) {
    for (int i = 0; i < points.size(); i++) {
        float sum = 0;
        for (int j = -SMOOTHING_FILTER_SIZE/2; j < SMOOTHING_FILTER_SIZE/2; j++) {
            int newIndex = i + j;
            if (newIndex >= 0 && newIndex < points.size()) {
                sum += points.get(i + j);
            }
        }
        sum /= SMOOTHING_FILTER_SIZE;
        points.set(i, (int) sum);
    }
}
```

Die Konstante `SMOOTHING_FILTER_SIZE` muss in der gleichen Klasse angelegt werden, in der die Methode eingesetzt wird.

Landeplattform: Damit der Lander bequem landen kann, müssen die Rechtecke mindestens so breit sein wie dieser. Um erfolgreich zu landen, muss der Lander komplett auf dem Rechteck ankommen.

Berührt der Lander andere Teile der Mondoberfläche, ist das Spiel verloren.

Lander

Dem Lander sollen Spieler über die Pfeiltasten Lenkimpulse geben können, während er konstant nach unten gezogen wird, um die Schwerkraft zu simulieren. Probieren Sie aus, mit welcher Kombination aus Impulsstärke und Schwerkraft das Spiel gut spielbar wird. Im Ordner `data\assets` liegt ein Bild, welches Sie für den Lander verwenden können.

Lenkimpuls: Nach Drücken der entsprechenden Pfeiltaste soll der Lander beschleunigt werden. Über die nächsten Wiederholungen des `draw()`-Loops in `LunarLander` soll diese Beschleunigung wieder abnehmen. Im zur Verfügung gestellten Video ist zu sehen, wie das Lenkverhalten aussehen soll. Achten Sie darauf, eine Maximalgeschwindigkeit einzurichten, damit das Verhalten realistischer wirkt. Der Lander darf das Spielfeld nicht verlassen.

Jeder Lenkimpuls verbraucht Treibstoff. Ist aller Treibstoff verbraucht, bevor der Lander gelandet ist, ist das Spiel verloren.

Treibstoffanzeige

Der zur Verfügung stehende Treibstoff sollte als Balken links oben im Spielfeld angezeigt werden, und bei jedem Lenkimpuls proportional verkleinert werden.

Vorgehen

Die gestellte Aufgabe ist nicht trivial. Der Schlüssel zur Lösung liegt in einer sorgfältigen Analyse des Problems, der sinnvollen Strukturierung des Programms und einer korrekten Behandlung der Steuerung und Kollision mit der Oberfläche.

Achten Sie bei der Modellierung des Landers darauf, dass die wirkenden Impulse sinnvoll modelliert werden, so dass auch ein lineares Abklingen gut umgesetzt werden kann. In jedem Schleifendurchlauf muss geprüft werden, ob der Lander mit einem Teil der Landschaft kollidiert.