Template

Zuletzt bearbeitet von Jürgen Hahn, Martina Emmert und Alexander Bazo

Template für Übungsaufgaben

Allgemeine Hinweise zur Studienleistung

In dieser Studienleistung werden Sie mehrere Programmieraufgaben lösen. Um die Aufgaben zu bearbeiten, müssen Sie zuerst das Projekt OOP-Studienleistung-WS20-01-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.

Starterpaket

Ein vorbereitetes Starterpaket zur selbständigen Implementierung der Aufgabe finden Sie hier.

Aufgabe 1: Bouncer räumt auf

In der ersten Aufgabe soll Bouncer Chaos beseitigen, indem er die bunten Kacheln aufsammel und nach Farbe sortiert.
Dabei soll Bouncer die Karte ablaufen und prüfen, ob er auf einem farbigen Feld steht und dies gegebenenfalls weiß einfärben.

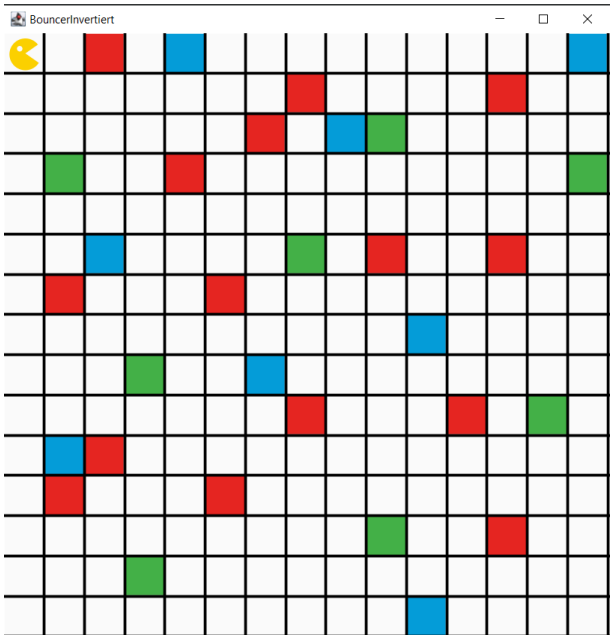

Abbildung 1: Dieses Chaos putzt Bouncer.

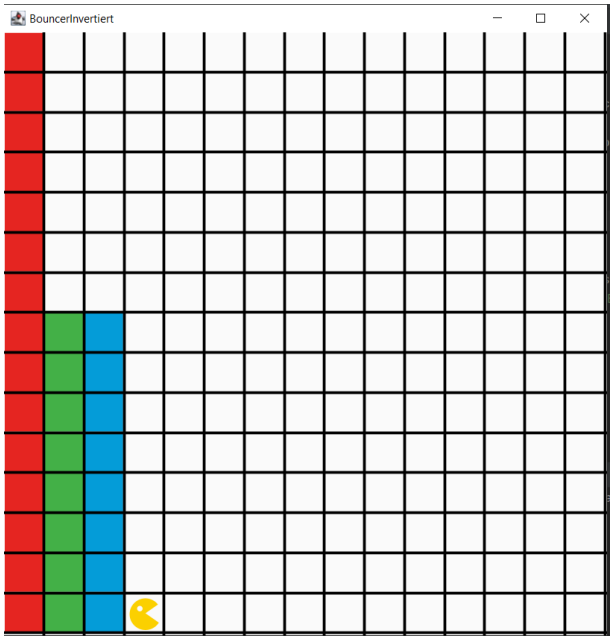

Abbildung 2: Bouncers Endergebnis.

Am Ende sollen die gesammelten Kacheln sortiert am unteren Ende der Karte wieder aufgestapelt werden. Hierfür zählt Bouncer - während er die Kacheln einsammelt - für jede der drei Farben mit, wie viele er eingesammelt hat.

Achten Sie darauf, dass Ihre Lösung für beliebig auf der Karte verteilte Kacheln gilt. Sie können überprüfen, ob Ihre Lösung dieser Vorgabe entspricht, indem Sie die im Starterpaket mitgelieferte zweite Karte Mess2.xml laden und mit Ihrem Code testen.

Hilfsmethoden: Legen Sie eigenständig Methoden an, die Ihnen dabei helfen, häufig auftretende Bewegungsabläufe von Bouncer auszulagern.

Aufgabe 2: Bouncer gräbt Tunnel

Die zweite Aufgabe ist anspruchsvoller: am unteren Rand der Karte ist ein wichtiger Rohstoff auf einem roten Feld deponiert. Bouncer muss sich durch die blaue Erde graben und dabei hartem Felsgestein ausweichen, um dorthin zu gelangen und den Rohstoff aufzusammeln indem er das Feld weissfärbt.
Finden Sie einen Weg, wie Sie Bouncer nach unten steuern können. Dabei müssen Sie darauf achten, dass der Rohstoff auf einem beliebigen Feld am unteren Rand liegen kann. Die Hindernisse am unteren Rand können maximal ein Feld hoch sein.
Bouncer sollte diese Aufgabe für verschiedene Karten lösen können. Sie können überprüfen, ob Ihre Lösung dieser Vorgabe entspricht, indem Sie die im Starterpaket mitgelieferte zweite Karte Tunnel2.xml laden und mit Ihrem Code testen. Folgende Annamen gelten: - Die Karte ist immer in zwei Bereiche getrennt: Luft (Weisse Felder) und Erde (blaue Felder). - Innerhalb der blauen Felder können Hindernisse (schwarze Felder) liegen - Diese liegen nie an der Oberfläche. - Die Hindernisse haben immer einen Abstand von mind. einem Feld untereinander. - Sollten Hindernisse am unteren Rand liegen, sind diese maximal ein Feld groß. - Der zu findende Rohstoff (rotes Feld) ist immer ein Feld groß und liegt immer am unteren Rand der Karte.

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 Abarbeitung der unterschiedlichen Fälle die bei einer Entscheidung während der Wegfindung nötig sind. Wie bei den bisherigen Bouncer-Aufgaben wird es darauf ankommen, dass Sie die elementare Aufgabe identifizieren, die Bouncer solange wiederholen muss, bis das Programm einen bestimmten Endzustand (hier: Bouncer steht auf dem roten Feld und färbt es weiss) erreicht hat. Als Hilfestellung dienen die folgende Fragen, die Sie sich zu Beginn der Bearbeitung stellen sollten. - Wann ist das komplette Problem gelöst? - Wie sieht der zu erreichende Endzustand aus und wie lässt sich im Programm prüfen ob dieser Zustand eingetreten ist? - Was muss wiederholt erledigt werden um den Hindernissen auszuweichen? - Welche Informationen über die aktuelle Umgebung müssen eingeholt werden um eine sinnvolle Entscheidung zu treffen?

Beginnen Sie mit der Lösung eines kleinen Teilproblems, z.B. damit, Bouncer sich so lange dreht, bis er nach unten schaut. Nehmen Sie sich dann den nächsten Schritt vor: Bouncer bewegt sich so lange nach unten, bis er ein blaues Feld erreicht. Im weiteren Schritt färben Sie dieses Feld weiss und steigen weiter ab.
Es könnte sinnvoll sein, das Problem generell in zwei Schritte zu teilen: Zuerst an den unteren Rand gelangen, dann den Rohstoff suchen. Indem Sie sich auf kleine, gut durchführbare Teilprobleme konzentrieren, machen Sie das Gesamtproblem handhabbar.

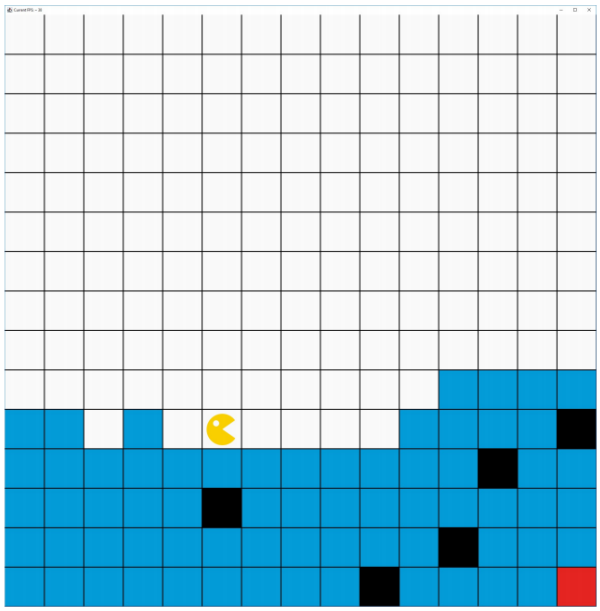

Abbildung 3: Bouncer als Tunnelgraeber.

Aufgabe 3: GraphicsApp - Kreisfahrt

Für die Lösung der 3. Aufgabe sollen Sie einen Kreis mit festem Radius im Uhrzeigersinn an den Rändern der Zeichenfläche entlang animieren. Der Kreis soll links oben starten und sich nach rechts bewegen, bis er an das Ende der Zeichenfläche stößt. Anschließend soll er die Richtung wechseln und sich am Rand entlang nach unten bewegen. Nach diesem Prinzip soll der Kreis den gesamten Rand der Zeichenfläche abfahren. Bei jedem Richtungswechsel soll der Kreis die Farbe zufällig wechseln. - Der Kreis darf sich nicht aus der Zeichenfläche heraus bewegen - In jeder Ecke bzw. bei jedem Richtungswechsel soll der Kreis die Farbe zufällig ändern - Die Animation soll kontinuierlich laufen, der Kreis soll sich also ständig im Uhrzeigersinn am Rand entlang bewegen - Achten Sie darauf, dass Sie den Kreis nur einmalig instantiieren

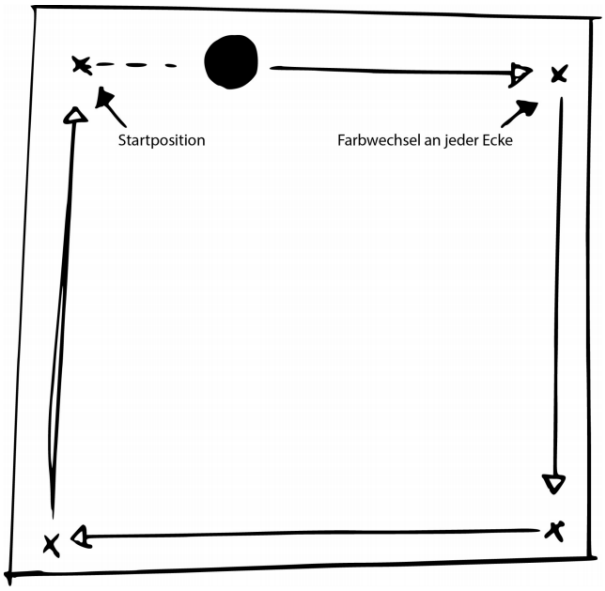

Abbildung 4: Skizze der Kreisbewegung.