Objektorientiertes Programmieren

Übungsbeispiele

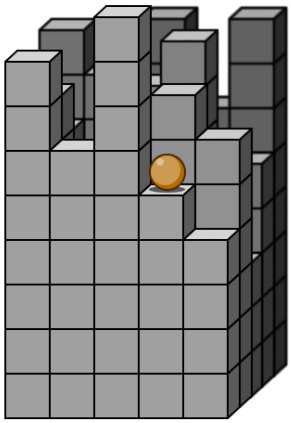

Figure 1: Visualisierung des Deep Miners (orange), der sich auf der Welt bewegt

1 Deep Miner

Implementieren Sie eine dreidimensionale Spielwelt mit einer Länge, Breite und Höhe von zumindest 5x5x10 (x-, y- und z-Koordinate). Diese Felder sollen zu Beginn zufällige Werte zwischen 1 und 9 erhalten. Danach soll es möglich sein, aus mehreren Minenrobotern einen zu wählen, mit dem man dann gegen den Computer rundenweise antritt und versucht, möglichst viele Punkte aus der Spielwelt abzubauen. Alternativ soll auch der Computer gegen sich selbst antreten können.
Folgende Punkte müssen dabei beachtet werden:

- Verwenden Sie für Datenstrukturen Klassen der STL.
- Verwenden Sie für Datenoperationen (suchen, sortieren, ...) Algorithmen der STL.

- Nutzen Sie Interfaces als Elternklassen und Polymorphismus für Ihre Objekte.
- Verwenden Sie sinnvolle Klassen. Beachten Sie dabei die richtige Verwendung von Zugriffsmodifikatoren, Gettern/Settern, den verschiedenen Konstruktoren und Destruktoren sowie der sinnvollen Strukturierung von Funktionen und Daten. Bilden Sie Ihre Klassen in einem entsprechenden Diagramm ab.
- Überprüfen Sie alle Parameterübergaben an Funktionen und Benutzereingaben auf Fehler und verhindern Sie so, dass Ihr Programm bei ungültigen Eingaben nicht mehr richtig funktioniert, Eingaben sollen so lange wiederholt werden, bis sie korrekt sind und der Spielfluss erst dann fortgesetzt werden.
- Testen Sie Ihren Code ausgiebig und berücksichtigen Sie Randbedingungen.

Stufe 1

Jede Runde soll daraus bestehen, dass man eine Richtung angibt, in die sich der eigene Roboter bewegen soll (er kann auch stehen bleiben), diese Bewegung ist nur entlang der x- und y-Achse möglich. Da der Roboter sich immer auf der Oberfläche der Spielwelt bewegen muss, ist die z-Koordinate immer gleich 9 minus der bereits abgebauten Felder der Spalte. Dann wird das derzeitige Feld, auf dem man steht, abgebaut, dazu wird der Wert der Spielwelt mit den derzeitigen Koordinaten des Roboters aus der Spielwelt entfernt und zur Gesamtpunktezahl des Roboters addiert. Implementieren Sie zumindest drei verschiedene Roboter, die sich durch ihre Abbau-Funktion unterscheiden, zwei sind hier vorgegeben, überlegen Sie sich einen eigenen dritten:

- Der Roboter sortiert alle z-Werte seiner derzeitigen x- und y-Koordinate in absteigender Reihenfolge und baut dann den Wert seiner Koordinaten ab (also den höchsten Wert der Reihenfolge).
- Der Roboter nimmt sich den Wert seiner Koordinaten sowie der nächsten zwei z-Werte.

Stufe 2

Immer wenn ein Roboter ein Vielfaches von 50 Punkten erreicht, wird die gesamte Spielwelt neu angeordnet. Dabei wird für jedes Paar aus x- und y-Koordinaten eine der folgenden Operationen zufällig gewählt und damit alle z-Werte dieser Koordinaten neu angeordnet:

- Die z-Werte werden zufällig gemischt.
- Die z-Werte werden aufsteigend sortiert.
- Die z-Werte werden absteigend sortiert.

Stufe 3

Erweitern Sie die Weltgeneration um Effektwerte. Diese Werte können nicht abgebaut werden, stattdessen wird vor dem Abbau von Werten überprüft, ob sich so ein Wert an irgendeiner z-Koordinate der derzeitigen x- und y-Koordinaten des Roboters befindet. Je nachdem, welcher Wert gefunden wird, passiert etwas anderes. Wird ein Effektwert ausgelöst, wird er danach aus der Welt entfernt und durch eine 0 ersetzt. Implementieren Sie folgende zwei Werte und überlegen Sie sich noch zumindest einen dritten:

- -1: Der Roboter darf diese Runde keinen Wert abbauen.
- -2: Der Roboter wird an die Stelle gesetzt, die den derzeit niedrigsten Wert bei $z = 9$ hat.

Anforderungen

- Fehlerüberprüfung der Inputs
- Erstellung eines UML-Diagramms, welches die Klassenstruktur darstellt
- Stufe 1
  - Generierung einer zufällig generierten 3D-Spielwelt in einen STL-Container
  - Implementierung von mindestens 3 Roboter-Typen
  - Implementierung der Bewegungs- und Abbaulogik
  - Darstellung der Spielwelt im Konsolenoutput
- Stufe 2
  - Neuanordnung der Spielwelt, sobald ein Vielfaches von 50 Punkten (50, 100, 150, ...) erreicht wurde
- Stufe 3
  - Implementierung von Effektwerten (-1, -2)

Bewertung

Aspekt	**Bewertung**
STL-Datenstrukturen & -Algorithmen	10%
Klassendiagramm	10%
Fehlerprüfung	10%
Stufe 1	20%
Stufe 2	30%
Stufe 3	20%