Übungsblatt 6

Abgabe: 30.11.2025

Die Abgabe wird ab jetzt auf *git* umgestellt, d.h. Abgaben auf anderen Wegen werden **nicht** mehr **akzeptiert**.

Aufgabe 1 Ein Ring, sie zu testen...

In Aufgabe 2 sollt ihr das Verfolgen eurer Spielfigur durch einen NPC implementieren. Eine geeignete Datenstruktur, um die dazu notwendige Zwischenspeicherung von Schritten zu realisieren, ist ein *Ringpuffer*. Da die Funktionsweise eines Ringpuffers nicht ganz einfach ist, soll er vorab getrennt vom Spiel getestet werden. Öffnet dazu das Projekt *RingBuffer* mit BlueJ. Die Klasse *RingBuffer* enthält bereits eine vermutlich korrekte Implementierung eines solchen Puffers.

Aufgabe 1.1 Testfälle definieren (60 %)

Überlegt euch eine Reihe von Tests, mit der die Implementierung des Ringpuffers getestet werden soll, damit ihr sicherstellt, dass sie sich auch so verhält, wie es die Kommentare im Quelltext beschreiben.[1] Erzeugt mit BlueJ eine Testklasse für *RingBuffer* und zeichnet die Tests auf oder implementiert sie von Hand.

Aufgabe 1.2 Teststärke prüfen (20 %)

Da die Implementierung vermutlich richtig ist, sollten alle Tests erfolgreich durchlaufen. Aber hätten eure Tests eigentlich Fehler gefunden? Um dies zu prüfen, nehmt mindestens 10 sinnvoll gewählte Änderungen an der Implementierung vor (immer nur eine zur Zeit), lasst die Tests jeweils durchlaufen und haltet fest, welche Tests nun fehlschlagen. Wenn eine Änderung unbemerkt bleibt, solltet ihr Aufgabe 1.1 noch einmal überarbeiten oder euch überlegen, warum das richtig sein kann. Änderungen könnten u.a. sein:

- Bedingungen auf *true* oder *false* setzen.
- Operatoren durch ähnliche ersetzen, z.B. > durch >=.
- Teile von Ausdrücken oder ganze Anweisungen weglassen.

Aufgabe 2 Spürhund (20 %)

Benutzt den Ringpuffer aus Aufgabe 1, um das Verfolgen eurer Spielfigur durch einen NPC zu realisieren. Erweitert die Implementierung aus Übungsblatt 3 so (oder erstellt eine neue Klasse), dass der NPC in einen Verfolgermodus umschaltet, wenn die Spielfigur in einer bestimmten Entfernung vor ihm platziert wird. Ein Ringpuffer wird genutzt, um die Bewegungen der Spielfigur aufzuzeichnen. Da diese nur Einerschritte macht und dabei in Richtung ihrer Schritte blickt,

[1] Verhalten, das als *undefiniert* angegeben ist, kann nicht getestet werden.

reicht es, ihre Rotationen zu speichern. Tragt erst die Rotation des NPC so oft in den Ringpuffer ein, wie es nötig ist, um durch Schritte die jetzige Position der Spielfigur zu erreichen. Tragt danach mit jedem weiteren Schritt der Spielfigur deren Rotation nach dem Schritt in den Ringpuffer ein. Zusätzlich entnehmt ihr immer eine Rotation aus dem Ringpuffer und bewegt euren NPC in diese Richtung. Dadurch führt der NPC zeitverzögert dieselben Bewegungen wie die Spielfigur aus.

Abgabe: Richtet unter *gitlab.informatik.uni-bremen.de* ein privates (!) Repository *pi1-2025*[2] ein und ladet eure Tutor:in dazu als *Developer* ein. Legt dort bereits auf dem Server eine geeignete *.gitignore*-Datei im Hauptverzeichnis des Repositories an. Klont das Repository und legt in eurer Arbeitskopie einen Ordner *loesung06*[2] an, in dem ihr eure Abgabe ablegt. *Commit*-tet eure Abgabe und *push*-t sie auf den GitLab-Server. Überprüft auf dem Server, ob eure Abgabe auch wirklich dort angekommen ist. Die Abgabe soll das enthalten, was bisher die zur Abgabe genutzten Archive enthalten haben (nun aber nicht mehr in Form eines Archivs), aber eben nicht die Dateien, die durch die *.gitignore*-Datei ausgeschlossen werden. Bitte beachtet, dass das GitLab der Uni häufiger kurz nach 23 Uhr für einige Minuten nicht erreichbar ist.

[2] Auch wenn viele von euch in Bremen zur Schule gegangen sind: Es ist exakt diese Zeichenfolge in genau dieser Groß- und Kleinschreibung gemeint.