Institut für Angewandte und Numerische Mathematik
Sommersemester 2017
Dr. Daniel Weiß, Dipl.-Math. Johannes Ernesti

Einführung in Python **Abschlussprojekt**

Aufgabe (Ein Spiel – Tower-Defense)

Ziel dieses Projektes ist die Entwicklung eines Computerspiels, bei dem eine fiktive Stadt vor angreifenden Räubern verteidigt werden muss.

Das Spiel findet in einer zweidimensionalen Spielwelt statt (vgl. Abbildung 1), in der sich die Angreifer auf einem im Voraus festgelegten Weg der Stadt nähern. Die Aufgabe des Spielers ist es, die Angriffe durch den Bau von Verteidigungstürmen abzuwehren, die er in bestimmten Bereichen der Spielwelt abseits des Wegs positionieren kann.

Abbildung 1: Ein Beispiel aus dem kommerziellen Spiel *Bloons TD 4*, bei dem die Angreifer die Form von Ballons haben und die Stadt links der sichtbaren Spielwelt liegt. Oben rechts sieht man die aktuelle Runde, das Vermögen des Spielers und die noch verbleibende Kraft der Stadtmauern. Außerdem gibt es am rechten Rand Schaltflächen, um neue Türme zu bauen.

Das Spiel läuft dabei in Runden ab, die alle gleich verlaufen. Die Runde beginnt damit, dass eine Menge von gleichartigen Angreifern am Beginn des Weges auf dem Spielfeld erscheint und sich dem Weg entlang der Stadt nähert. Befindet sich ein Angreifer im kreisförmigen Aktionsradius eines Turmes, so attackiert der Turm den Angreifer, wodurch dieser Schaden erleidet. Jeder Angreifer hält nur ein gewisses Maß an Schaden aus, sodass sofort aus dem Spiel entfernt wird, sobald ihm die Türme entsprechend zugesetzt haben. Schafft es ein Angreifer, die Stadt zu erreichen, scheidet er ebenfalls aus dem Spiel aus und die Stadt erhält einen Schadenspunkt.

Befinden sich keine Angreifer mehr auf dem Spielfeld, weil sie entweder durch die Türme eliminiert worden sind oder die Stadt erreicht haben, endet die Runde und die nächste Welle von Angreifern macht sich bereit.

Der Spieler erhält für jeden vernichteten Angreifer ein gewisse Summe Spielgeld, die er verwenden kann, um weitere Türme zu bauen.

Jedes Spiel hat eine im Vorhinein festgelegte Anzahl solcher Spielrunden, wobei das Spielziel darin besteht, die Stadt gegen die Angreifer aller Runden zu verteidigen, d.h. die erlittenen Schadenspunkte der Stadt über das ganze Spiel zu minimieren. Der Spieler verliert das Spiel, wenn die Angreifer der Stadt so viel Schaden zuführen, dass ihre Mauern brechen.

1 Modellierung der Spielsituation

(a) Modellieren Sie die das Angreifer und Türme durch geeignete Klassen. Jeder Angreifer hat einen Namen, eine Fortbewegungsgeschwindigkeit, einen Wert in Spielgeld und eine Lebensenergie. Ein Turm besitzt ebenfalls einen Namen, eine Reichweite, eine Feuerrate, einen Preis in Spielgeld und den Schaden, den ein Treffer beim getroffenen Angreifer anrichtet.

(b) Überlegen Sie sich ein möglichst einfaches Dateiformat, um ein Spielfeld zu beschreiben. Dabei sollen der Weg der Angreifer, die Position der Stadt und ggf. schon zu Beginn des Spiels vorhandene Türme beschrieben werden können. Der Weg soll dabei aus nummerierten Wegpunkten bestehen, welche die Angreifer der Reihe nach auf geraden Laufwegen passieren.

Hinweis: Eine gute Möglichkeit besteht darin, das Spielfeld aus rechteckigen Kacheln gleicher Größe zusammenzusetzen. Beispielsweise gäbe es dann Kacheln für Teile des Weges, das Stadttor und sonstige Teile des Spielfelds.

(c) Modellieren Sie das Spielfeld durch eine geeignete Klasse, die insbesondere Methoden zum Auslesen des Spielfeldes aus einer Datei bereitstellt.

(d) Überlegen Sie sich ein Dateiformat, um alle Wellen von Angreifern eines Spiels zu beschreiben. Die Datei soll die Eigenschaften und Anzahlen der Angreifer jeder Runde spezifizieren.

Erweitern Sie das Dateiformat so, dass zusätzlich die verfügbaren Türme und die Robustheit der Stadt beschrieben werden können.

(e) Implementieren Sie eine Klasse, welche die Konfiguration eines Spiels modelliert. Dabei sollen das Spielfeld, die Beschreibung der Angreifer und die verfügbaren Türme aus Dateien eingelesen werden.

2 Die Spielmechanik

Das Spiel soll in einer diskreten Zeit ablaufen. Dazu wird eine Zeitschrittweite Δt gewählt der Spielverlauf durch den Zustand der Stadt, Türme und Angreifer zu den diskreten Zeitpunkten $0, \Delta t, 2\Delta t, 3\Delta t, \ldots$ beschrieben.

Alle Interaktionen zwischen den Angreifern, dem Spielfeld, der Stadt und den Türmen finden ausschließlich zu diesen diskreten Zeitpunkten statt. Ein Angreifer der Geschwindigkeit v *springt* also zwischen zwei aufeinander folgenden Zeitpunkten um die Strecke $v\Delta t$ auf dem Spielfeld. Ein Turm, der n mal pro Zeiteinheit angreifen kann, verteilt zu jedem diskreten Zeitpunkt $k\Delta t$ insgesamt n Schüsse, zufällig verteilt auf alle Angreifer, die sich zu diesem Zeitpunkt in seinem Aktionsradius befinden. Dabei sollen die Flugzeiten der Geschosse vernachlässigt werden, sodass sofort nach dem Abschuss der Schaden beim Angreifer entsteht.

(a) Schreiben Sie eine Klasse, deren Instanzen den Zustand eines Spiels zu jedem diskreten Zeitpunkt $k\Delta t$ vollständig beschreibt.

Implementieren Sie insbesondere einen Konstruktor, der Ihre Ergebnisse aus den vorherigen Aufgaben verwendet, um einen konsistenten Anfangszustand herzustellen.

Hinweis: Wenn Sie das Spielfeld aus Kacheln gleicher Größe aufgebaut haben, können Sie beispielsweise festlegen, dass nur ein Turm auf einer Kachel stehen kann, sodass die Positionen der Türme im diskreten Kachel-Koordinatensystem festgelegt werden. Für die Angreifer kann es sinnvoll sein, Gleitkomma-Koordinaten zu verwenden, sodass die Angreifer auf einer feineren Skala positioniert werden können. Damit nicht alle Angreifer "übereinander" über das Spielfeld laufen, können sie so beispielsweise zufällig in einem Startbereich verteilt werden.

(b) Erweitern Sie die Klasse um eine Methode, die den Übergang des Spiels von einem diskreten Zeitpunkt zum nächsten durchführt.

Berücksichtigen Sie dabei die Bewegung der Angreifer, die Schüsse der Türme und das Ausscheiden der Angreifer aus dem Spiel. Sie können jegliche Art der Kollisionserkennung zwischen den Angreifern untereinander sowie zwischen Angreifern und Türmen vernachlässigen.

(c) Versehen Sie die Methode aus (b) mit Ausgaben, die den Spielverlauf dokumentieren, und testen Sie Ihr Spiel an einfachen Beispielkonfigurationen, wobei Sie die Methode aus (b) immer wieder erneut aufrufen.

3 Grafische Ausgabe und Benutzereingaben

Die bisher entwickelten Programmteile liefern schon ein funktionsfähiges Spiel, das die Gestaltung von verschiedenen Spielsituationen erlaubt. Um das Spiel zu vervollständigen fehlen aber noch zwei wesentliche Punkte:

- eine graphische Visualisierung des Spielgeschehens
- Benutzerinteraktion, um beispielsweise neue Türme zu bauen

Die folgenden Aufgaben schließen diese Lücken.

(a) Gestalten Sie mithilfe von `PyQt` eine graphische Oberfläche für Ihr Spiel, die mithilfe von geeigneten Steuerelementen Informationen zum aktuellen Zustand des Spiels bereitstellen kann, d.h. Zustand der Stadt, Nummer der aktuellen Angreiferwelle, Vermögen des Spielers...

Reservieren sie im Spielfenster einen großen Bereich, um die zweidimensionale Spielwelt inklusive aller momentan dort befindlichen Angreifer und Türme darzustellen, wobei dieser zunächst ohne Funktion sein soll.

(b) Verwenden Sie die bisher entwickelten Programmteile, um ein Programm zu erstellen, das ein Spiel initialisiert und Ihre Oberfläche aus Aufgabenteil (a) anzeigt.

(c) Füllen Sie die reservierte Fläche aus (a) mit einem geeigneten Steuerelement, mit dem sich Grafiken ausgeben lassen. Implementieren Sie eine Methode, die auf diesem Steuerelement das Spielfeld in seiner aktuellen Konfiguration zeichnen kann. Sie können dabei wahlweise Primitive wie Kreise, Rechtecke etc. oder Bilddateien für die Darstellung der Spielelemente verwenden.

Hinweis: Informieren Sie sich über `QPainter` und ggf. `QGraphicsView`.

(d) Fügen Sie Ihrer Oberfläche ein `QTimer`-Steuerelement hinzu, das bei jedem `timeout`-Signal den Spielzustand aktualisiert und die Visualisierung neu zeichnet. Das Attribut `intervall` dieses Steuerelements entspricht dabei dem Δt von oben.

Hinweis: Durch eine hinreichend kleine Wahl von Δt kann damit der Eindruck eines kontinuierlichen Spielverlaufs erweckt werden.

(e) Erweitern Sie die Oberfläche um weitere Schaltflächen, mit deren Hilfe neue Türme gebaut werden können. Dazu wählt der Spieler einen Turm zunächst durch Klick auf die Schaltfläche aus und positioniert ihn mit einem weiteren Klick auf die Visualisierung des Spielfelds.