technische universität
dortmund

Fakultät für Informatik
Lehrstuhl 14
Lars Hildebrand

Praktikum zu

Einführung in die Informatik für LogWilngs und WiMas

Wintersemester 2018/19

Tower-Defense-Übungsblatt

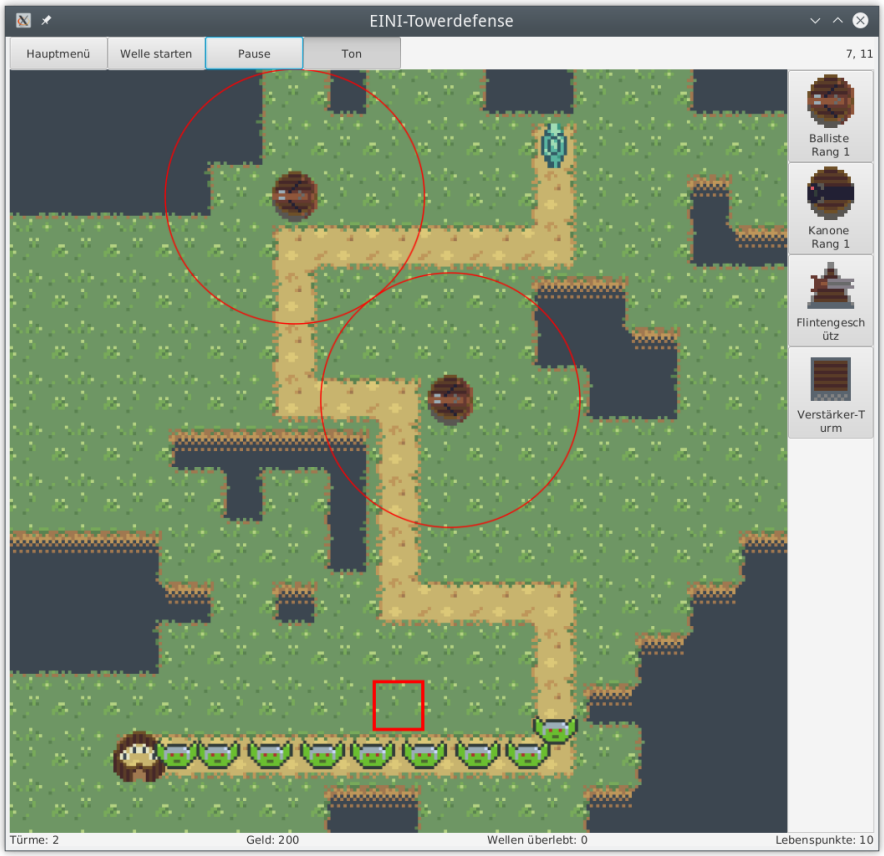
Was ist ein Tower-Defense-Spiel?

Bei einem Tower-Defense-Spiel durchqueren Gegner das Spielfeld. Ziel des Spiels ist es, diese Gegner daran zu hindern. Um die Gegner daran zu hindern, kann der Spieler Türme errichten, die den Gegnern Schaden zufügen. Die Gegner erscheinen in sogenannten Wellen, die vom Spieler gestartet werden. Getötete Gegner und überstandene Wellen bringen Geld, von dem weitere Türme gebaut werden können. Erreichen zu viele Gegner ihr Ziel, hat der Spieler verloren.

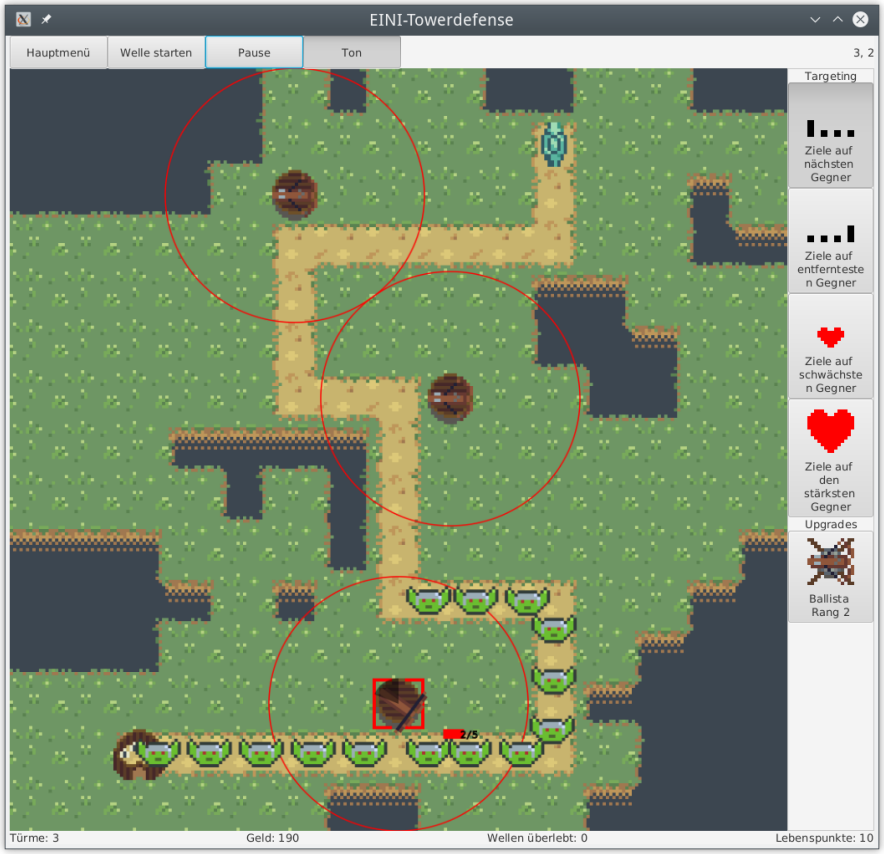
Wie spielt man unser Spiel?

Unsere Gegner sind Goblins (). Sie starten an ihrem Startpunkt () und bewegen sich über einen Pfad zu ihrem Zielpunkt (). Wenn 10 Goblins ihr Ziel erreichen, ist das Spiel vorbei.

Das Spielfeld ist eingeteilt in Kacheln. Auf bebaubare Kacheln () lassen sich Türme bauen, die selbstständig Goblins angreifen, die sich in ihrem Angriffsradius befinden. Der Angriffsradius wird als roter Kreis dargestellt.



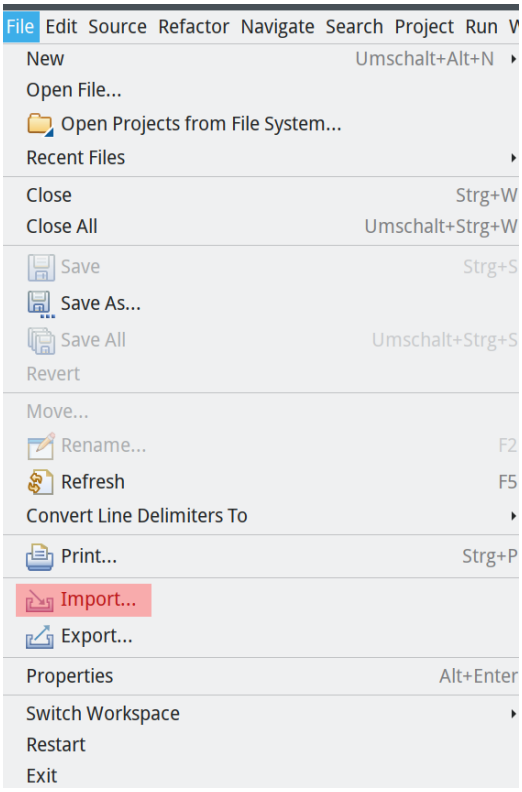
Klicken Sie auf eine Kachel ohne Turm, werden rechts die Türme angezeigt, die gebaut werden können. Klicken Sie auf eine Kachel mit Turm, werden rechts die Zielstrategien angezeigt, mit denen die Türme ihr nächstes Ziel auswählen, sowie eventuell verfügbare Upgrades des Turms.



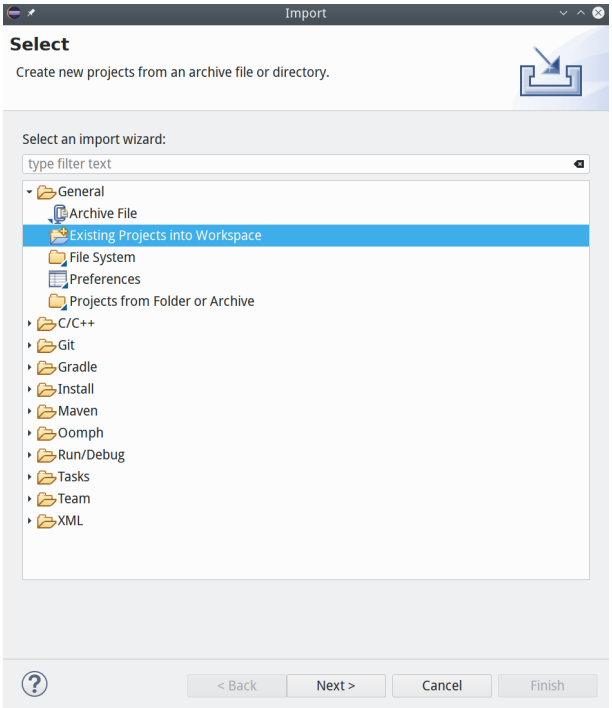
Einige Dinge werden erst funktionieren, wenn die entsprechende Programmieraufgabe gelöst wurde. Die Zielstrategien, um auf den stärksten oder schwächsten Gegner zu zielen, gehören auch dazu.

Wie importiert man die Übungsaufgaben?

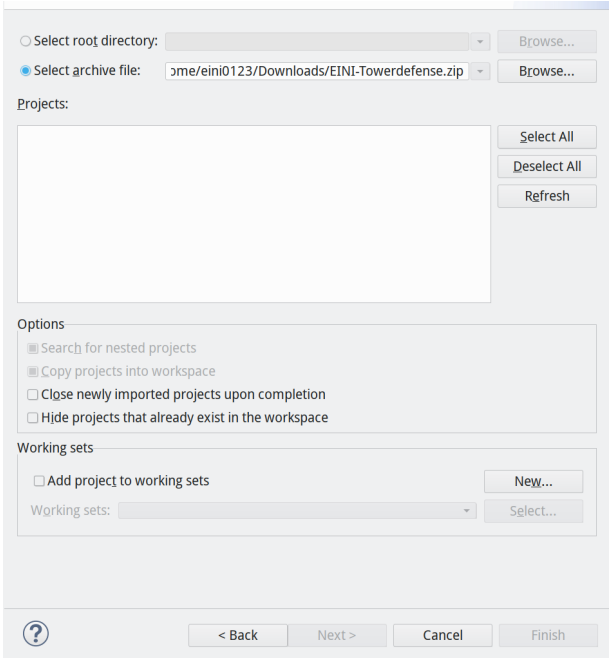
- Laden Sie das Quellcode-Archiv von der EINI-Seite herunter. Dieses sollen Sie **nicht** manuell entpacken.
- In Eclipse klicken Sie auf *File*, danach auf *Import...*



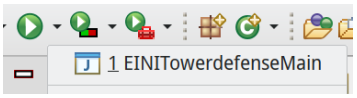
- Im sich daraufhin öffnenden Fenser wählen Sie *General* und danach *Existing Projects into Workspace* aus. Klicken Sie danach auf *Next*.



- Klicken Sie auf *Select archive file* und suchen Sie die Archivdatei heraus.



- Stellen Sie sicher, dass ein Haken bei *Copy projects into workspace* gesetzt ist. Klicken Sie danach auf *Finish*.
- Bis auf Aufgabe 6 muss ausschließlich im Paket **aufgaben** in der jeweiligen Klasse programmiert werden.
- Sie starten das Spiel, indem Sie die Klasse **EINITowerdefenseMain** im Paket **view** starten. Sie können sicherstellen, die richtige Datei zu starten, indem Sie auf das Dreieck neben dem grünen Play-Button drücken und die richtige Klasse auswählen.



Aufgaben

Aufgabe 1: Abstand zweier Punkte

Um die grundlegende Funktionalität des Spiels herzustellen, implementieren Sie die Funktion **distance**, die den Abstand zweier Punkte berechnet. Die Parameter **firstX** und **firstY** sind die X- und Y-Koordinate des ersten Punktes, **secondX** und **secondY** die Koordinaten des zweiten Punktes. Diese Funktion wird vor allem dafür verwendet, um festzustellen, welche Gegner in Reichweite sind.

Aufgabe 2: Türme zählen

Die Anzeige links unten in der Spielfeldansicht soll die Anzahl der gebauten Türme anzeigen. Schreiben Sie die Funktion **numberOfTowers**, die die Anzahl der Türme auf dem Spielfeld zählt.

Für diese Aufgabe haben Sie mehrere Hilfsfunktionen gegeben. Achten Sie auf die Grenzen des Spielfeldes! Bekommen Sie eine NullPointerException oder IndexOutOfBoundsException, versuchen Sie wahrscheinlich, auf nicht vorhandene Kacheln zuzugreifen.

Aufgabe 3: Schwächsten/Stärksten Gegner finden

Die Zielstrategien, die den stärksten bzw. schwächsten Gegner finden sollen, rufen die Funktionen **indexOfWeakest** und **indexOfStrongest** auf. Implementieren Sie diese beiden Funktionen. Sie

erhalten ein Array mit den Lebenspunkten der Gegner in Reichweite eines Turms und sollen den **Index** mit dem größten bzw. kleinsten Wert zurückgeben.

Aufgabe 4: Türme zählen – um eine Koordinate herum

Die Verstärkertürme machen die umliegenden Türme stärker. Steht ein einzelner Turm neben einem Verstärkerturm, wird er doppelt so stark, zwei Türme werden $1\frac{1}{2}$ mal so stark, drei Türme $1\frac{1}{3}$ mal so stark, etc.

Damit dies funktioniert, muss die Funktion **countTowersAround** implementiert werden. Sie können davon ausgehen, dass auf dem Feld in der Mitte ein Turm steht, der **nicht** mitgezählt werden soll. Achten Sie auch hier wieder auf die Grenzen des Spielfeldes.

Aufgabe 5: Pfad für den teleportierenden Gegner erstellen

Der Magiergoblin verfolgt den Pfad nicht wie die anderen Gegner. Er teleportiert sich auf zufällige Kacheln auf dem Pfad und die Anzahl seiner Schritte ist die Anzahl der Kacheln auf dem Pfad.

Verwenden Sie Ihre Lösung zur Aufgabe zur verketteten Liste und kopieren Sie sie in das Paket **aufgabenrahmen**. Implementieren Sie die Funktion **randomPath**, ihr Parameter ist die Länge des Pfades. Sie soll eine verkettete Liste aus zufälligen Kachel-Indices zurückgeben, die so lang ist wie der Parameter **length**. Das letzte Element der Liste muss jedoch auf jeden Fall `length-1` sein, damit der Gegner sein Ziel erreicht!

Aufgabe 6: Erstellen einer neuen Turmklasse

Wenn Sie das Spielfeld zur Aufgabe betrachten, werden Sie erkennen, dass kein Turm weit genug schießen kann, um die Gegner zu erreichen. Deswegen sollen Sie eine Turm-Klasse mit größerer Reichweite implementieren.

- Legen Sie sie im Paket **model.towers** an und lassen Sie sie von der Klasse **BallistaTowerI** erben.
- Rufen Sie im Konstruktor den Konstruktor der Oberklasse (ohne Parameter) auf und setzen das **range**-Attribut auf einen höheren Wert, z. B. 10.
- Überschreiben Sie die Methode **getName**, um im Auswahlmenü Ihren Turm leichter zu finden.
- Fügen Sie in der Methode **getBuildableTowers** der Klasse **Session** im Paket **model** Ihre Turm-Klasse zur Liste hinzu.