Universität Heidelberg
Dr. Daniel Kondermann

Aufgabenblatt 7

Für dieses Aufgabenblatt haben Sie zwei Wochen Bearbeitungszeit und können daher **20 Punkte** erreichen. Für freiwillige sinnvolle Erweiterungen des Grundspiels können Sie auf diesem Aufgabenblatt weitere Bonuspunkte sammeln.
Packen Sie Ihre komplette Visual Studio Solution in **einem ZIP-Archiv**, das sie nach dem Muster '*SheetX-Surname.zip*' benennen und laden sie dieses im Moodle hoch

---

Sie haben bereits zwei Spiele programmiert – Pong und Snake im prozeduralen Stil, sowie Snake im OOP-Stil. Auf diesem Aufgabenblatt sollen Sie das Spiel Frogger im OOP-Stil implementieren. Zur Visualisierung und Steuerung des Spiels verwenden Sie wieder die Klassen *Terminal* und *Vec2D*, die Sie bereits von vorherigen Aufgabenblättern kennen. Wie üblich finden Sie ein Code-Template im Moodle.
Die Spielregeln für Frogger sind wie folgt: Sie sind ein Frosch und hüpfen über das Spielfeld, bis Sie zu einer Straße kommen, die Sie überqueren sollen, ohne dabei von den vorbeifahrenden Autos überfahren zu werden. Ein Beispiel für die grafische Umsetzung des Spiels sehen Sie auf folgendem Bild.

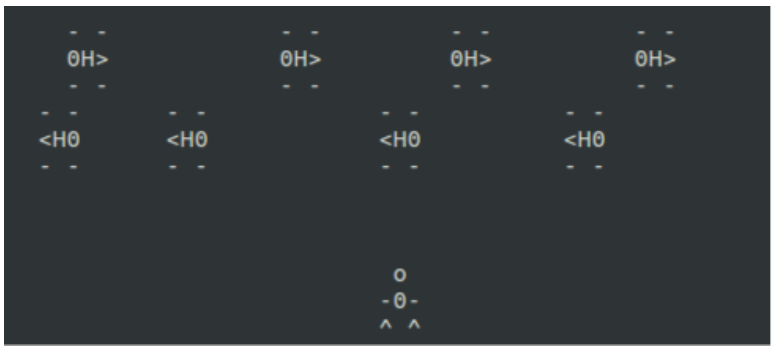

Abbildung 1: Frogger Grundspiel

Für ein besseres Verständnis des Spielprinzips und zum Sammeln von weiteren Ideen lassen Sie sich ruhig von YouTube Videos inspirieren.

Aufgabe 1 Klasse Car und Klasse Frog

Beginnen Sie zunächst damit eine Klasse *Frog* und eine Klasse *Car* zu implementieren, die nur einen Konstruktor und eine Methode besitzen, welche den Körper des Objekts auf dem Terminal zeichnet. Testen Sie zunächst das Zeichnen dieser Objekte auf dem Terminal in der main-Funktion.

Hinweis: Damit Sie auf dem Terminal zeichnen können, muss mindesten einmal die Funktion `char get_key()` der Klasse Terminal aufgerufen werden.

Tipp: Mehrere Objekte vom gleichen Typ lassen sich in einem `std::vector` zusammenfassen.

Aufgabe 2 Klasse Game

Erstellen Sie nun eine Klasse *Game*. Lassen Sie sich hierfür von der *Game*-Klasse des Snake-Spiels von Aufgabenblatt 5 inspirieren. *Game* soll nun alle im Spiel vorkommenden Objekte enthalten und den Spielablauf regeln, so dass in der main-Funktion nur noch ein Objekt von jeweils dem Typ Game und Terminal existiert.

Aufgabe 3 Methoden

Fügen Sie den Klassen *Car* und *Frog* Methoden hinzu, die folgendes ermöglichen: Der Frosch soll mittels Tasteneingabe nach vorne, links beziehungsweise rechts hüpfen können. Ein Auto soll sich von alleine entweder nach links oder rechts bewegen, je nachdem welche Richtung ihm zugewiesen wurde. Verändern Sie hierfür den existierenden Konstruktor oder erstellen Sie eine Methode, die die Fahrtrichtung zuweist. Weiterhin soll überprüft werden, ob der Frosch von einem Auto überfahren wird, womit das Spiel beendet wäre.

Aufgabe 4 Klasse Highway (Optional)

Sie können nach Belieben eine Klasse *Highway* erstellen, die mehrere Objekte vom Typ *Car* zusammenfasst und deren Dynamik regelt (Abstand zwischen den Autos, neues Auto hinzufügen etc.), oder der Klasse *Game* direkt mehrere Autos als Eigenschaft geben. Beachten Sie, dass Sie nicht nur neue Autos hinzufügen, sondern auch diejenigen Autos löschen, die das Spielfeld verlassen haben.