Fortgeschrittene objektorientierte Programmierung
LVA 185.211, VU, 3 ECTS, 2014 S

2. Übungsaufgabe

Lernen Sie Smalltalk und entwickeln Sie ein Programm in Smalltalk (siehe unten). Jedes Mitglied einer Übungsgruppe soll sich in etwa gleichem Maße an dieser Aufgabe beteiligen. Ziel der Aufgabe ist das Sammeln von Erfahrungen zur dynamischen objektorientierten Programmierung, aber auch konkret zur Programmierung in Smalltalk. Historisch gesehen hat Smalltalk nicht nur aktuelle Programmiersprachen und -paradigmen stark beeinflusst (viele wesentliche objektorientierte Sprachkonzepte), sondern auch die Art und Weise, wie wir heute mit Computern umgehen (Fenster, grafische Entwicklungsoberfläche, personalisiertes System). Es gibt mehrere frei verfügbare Smalltalk-Systeme. Prinzipiell sind alle davon zur Lösung der Aufgabe geeignet. Empfohlen wird das Smalltalk-System Squeak, das (im Vergleich zu einigen anderen, eher historisch interessanten Systemen) relativ neu ist und sich gut zum Programmieren von Spielen ohne besondere Vorkenntnisse eignet.

Squeak:

Squeak ist eine Open-Source-Implementierung von Smalltalk-80, die auf vielen Plattformen verfügbar ist. Details der Installation hängen von der Plattform ab. Beachten Sie, dass Sie neben dem eigentlichen Interpreter noch einige Dateien brauchen, in denen der aktuelle Systemzustand enthalten ist. Wenn Sie in Smalltalk programmieren, verändern Sie diese vorgegebenen Dateien, statt für Ihren Quellcode eigene Dateien anzulegen. Wenn Sie mehrere Versionen haben wollen, brauchen Sie auch mehrere solche vorgegebenen Dateien. Am wichtigsten ist die Datei “squeak.image”, die das gesamte System in einem ausführbaren Zwischencode enthält. Die Datei “squeak.changes” enhält alle Änderungen des Smalltalk-Codes und “SqueakV41.sources” (wahrscheinlich mit einer anderen einkodierten Versionsnummer) den Source-Code des Grundsystems. Wenn Sie “squeak” nicht in dem Verzeichnis aufrufen, in dem diese Dateien zu finden sind, müssen Sie die Dateien beim Aufruf explizit angeben.

Squeak unterstützt zwei grundsätzlich verschiedene Subsysteme zur Programmierung grafischer Oberflächen – das klassische “MVC” (Model-View-Controller, wie in Smalltalk-80) und das neuere “Morphic”. Sie können eines der beiden Subsysteme wählen, bevorzugt das neuere. Vermeiden Sie die gemischte Verwendung, bei der es zu unerwarteten Wechselwirkungen kommen kann.

Aufgabe:

Schreiben Sie eine neue Variante des Mäuserennpiels aus der 1. Aufgabe, wobei das ganze Spiel aber nur auf einem Rechner in einem Fenster laufen soll. Damit das Spiel in Smalltalk einen eigenständigen Charakter bekommt, soll der Spielablauf jedoch anders sein: Es gibt keine Türen, die von Spieler(innen) geöffnet oder geschlossen werden könnten. Stattdessen kann jede(r) Spieler(in) durch Druck einer Taste ihrer oder seiner Maus befehlen, ein Stück in eine zufällige, nicht durch eine Mauer versperrte Richtung zu rennen. Mit jedem Tastendruck wird eine neue zufällige Richtung gewählt. Wenn die Taste nicht gedrückt wurde oder die Wirkung des letzten Tastendrucks vorbei ist, läuft die Maus wieder geradewegs auf den Ausgang zu bzw. einer Mauer entlang, die sie näher an den Ausgang bringt. In einer Sackgasse bleibt die Maus einfach stehen. Wenn zwei Mäuse sich begegnen, beschnuppern sie sich und bleiben ebenso stehen. Aus diesen Situationen kann eine Maus nur durch Tastendruck befreit werden. Jeder Maus ist nur eine Taste zugeordnet. Durch einen einzelnen Tastendruck kann man zwar keine Richtung bestimmen, aber durch Wiederholen des Tastendrucks (falls die Maus in eine unerwünschte Richtung rennt) hat man dennoch Einfluss auf die Richtung. Die Tasten sind so zu wählen, dass sich die Spieler(innen) auf nur einer Tastatur möglichst wenig gegenseitig behindern. Die Reaktionszeit nach einem Tastendruck muss kurz sein um sinnvoll spielen zu können.

Wie die Aufgabe zu lösen ist:

In Smalltalk schreibt man in der Regel nicht einfach ein Programm für einen bestimmten Zweck, sondern man adaptiert und erweitert die vorhandene Umgebung. Ein Ziel der Aufgabe ist es, diesen Programmierstil kennenzulernen. Vor allem zur Programmierung der grafischen Oberfläche empfiehlt es sich, ein anderes Programm als Vorlage zu verwenden und dieses abzuändern. Dabei werden Sie lernen, sich im Smalltalksystem zurechtzufinden. Smalltalk-Programmierer(innen) lernen eher durch das Lesen und Verwenden bestehenden Smalltalk-Codes als aus der Dokumentation. Daher sind Kommentare besonders wichtig, und jede Klasse und Methode, die Sie selber schreiben, soll durch Kommentare klar spezifiziert sein.

Für Squeak gibt es unter anderem die Zusatzpakete “Etoys” und “Scratch”, die die Erstellung einiger Spiele stark vereinfachen. Zur erfolgreichen Lösung der Aufgabe ist es notwendig, dass Sie im eigentlichen Smalltalk-System Code schreiben bzw. adaptieren, also nicht einfach nur solche Zusatzpakete verwenden.

Die einfache Änderbarkeit des Systems kann sich auch negativ auswirken. Insbesondere wird es dadurch schwieriger, Programme von einem System auf ein anderes zu übertragen. Zur Verringerung dieses Problems wurden Versionsverwaltungssysteme wie z.B. Monticello eingeführt. Verwenden Sie ein solches System um das von Ihnen entwickelte Spiel portabel zu gestalten.

Ein Grundkonzept hinter Smalltalk nennt sicht “Personal-Computing” (PC), das heißt, jede(r) Benutzer(in) hat ein eigenes System – ein Begriff, der schon lange losgelöst von Smalltalk verwendet wird. Obwohl dieses Konzept sehr erfolgreich war und ist, stößt es bei der Zusammenarbeit mehrerer Personen gelegentlich an seine Grenzen. Um diese Problematik klarer zu machen (und aus offensichtlichen didaktischen Gründen) sollen alle Mitglieder jeder Übungsgruppe gemeinsam an der Lösung dieser Aufgabe arbeiten. Es reicht nicht, wenn nur ein Teil der Gruppe die Arbeit macht und dem Rest der Gruppe die Lösung erklärt.