Spezifikation der Semesterarbeit *

Aufgabenteil 1: Beschreibung

Ziel des Aufgabenteils ist die Programmierung eines Command Line Interfaces (CLI). Das Programm soll Informationen aus zwei unterschiedlichen Datenquellen auslesen, zwischenspeichern, durchsuchen und anschließend in ein neues Format überführen können. Eine der Datenquellen muss eine XML (mit zugehöriger DTD) sein. Das neue Format muss ebenfalls vom Typ XML sein und die Daten der Ursprungsdateien sinnvoll zusammenführen.
Inhaltlich können Sie sich frei bewegen: Es bleibt Ihnen / Ihrer Gruppe überlassen, ob Sie Personendaten und Geburtstage, Produktinformationen und Lagerbestände oder beliebige andere Informationen verarbeiten wollen. Die Semesterarbeit kann in 4 Stufen erfüllt werden. Sie können die in den Vorlesungsfolien beschriebenen Token– und Attributklassen anwenden, wenn Sie möchten. (Vgl.: http://hki.uni-koeln.de/cpp-ws1718-task/)

Hinweis: Grundsätzlich soll nach Möglichkeit ein objektorientiertes Programm umgesetzt werden.
Es empfiehlt sich eine Planung des Projektes vor seiner Implementierung.

Stufe I - Basisaufgabe

1. Entwerfen und Beschreiben Sie einen konkreten Anwendungsfall für Ihr Programm.
2. Legen Sie zwei Unterschiedliche Datenquellen an. Eine muss eine XML-Instanz mit DTD sein.
   Die andere ein CSV-Format. (Legen Sie die Datenquellen mit Blick auf Stufe IV an!)
3. Die Komplexität Ihrer Datensätze darf die der Beispiele des vergangenen Jahres nicht unterschreiten:
   http://hki.uni-koeln.de/cpp-ws1718-task/
   Hinweis: Wenn Sie auf Schwierigkeiten mit Ihren XML-Instanzen stoßen kann sich ein Blick in Ihre BSI-Unterlagen aus dem 1. Semester lohnen.
4. Lesen Sie die Daten mit Hilfe eines Parsers ein und ermöglichen Sie folgende Operationen:

4.1 Menüführung: Ein Menü startet verschiedene Abläufe im Programm und kehrt danach ins Menü zurück.
4.2 Konvertieren: Das Programm fragt nach Name und Pfad der Ausgabe und exportiert die Informationen aus beiden Datensätzen in einer neuen, vereinten XML Instanz, welche Sie inkl. DTD definieren.

Stufe II

Die Basisaufgabe wird erfüllt. Zusätzlich ermöglich das Programm
1. Alle (vereinten) Datensätze auszugeben (d.h. darzustellen). Alle Datensätze haben dabei eine Nr. / ID.
2. Neue Datensätze anzulegen. Die ID wird durch das Programm vergeben und darf nicht doppelt vorkommen.
3. Datensätze zu löschen. Die geschieht per Angabe der Nr. / ID

Stufe III

Stufe II wird erfüllt. Zusätzlich bietet das Programm die Möglichkeit, alle (vereinten) Datensätze sinnvoll anhand von 2 Kriterien zu durchsuchen. Dies könnte zB ein Name oder ein numerischer Wert sein, der als Eigenschaft des Datensatzes vorliegt. Die Suche gibt eine Liste aller Treffer aus. Erläutern Sie die Funktionalität der Suche bei der Einreichung **kurz**.

Stufe IV

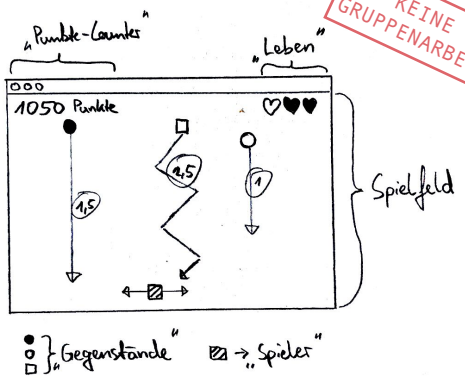
Stufe III wird erfüllt. Außerdem wird das Programm um folgende Operationen ergänzt:
1. Erweitern Sie die Software um eine Berechnung über mehrere Datensätze hinweg (zB Anzahl oder Werte addieren).
2. Ermöglichen Sie einen Export der berechneten Daten.
3. Erweitern Sie Ihr CLI um eine Prüfung der gegebenen XML-Instanz anhand Ihrer DTD.

GRUPPENARBEIT
BIS 3 PERS. MÖGLICH

Aufgabenteil 2: Beschreibung

KEINE GRUPPENARBEIT

Ziel des Aufgabenteils ist die Programmierung und Gestaltung eines graphischen Spiels unter Verwendung von C++ und QT.
Der Spieler soll eine kleine geometrische Figur mittels der Pfeiltasten rechts entlang einer Line am unteren Ende des Spielfeldes steuern können. Von der oberen Spielfeldkante treten „Gegenstände“ ins Spiel ein, die sich zum unteren Ende des Spielfelds bewegen. Ziel des Spiels ist das Ausweichen vor den fallenden Gegenständen bzw. das möglichst lange Überleben.
Der Aufgabeteil kann in 4 Stufen erfüllt werden. Es empfiehlt sich allerdings alle Stufen zu durchdenken, und ein Konzept zu entwickeln, bevor man mit der eignetlichen Implementierung beginnt.



Beispielhafte Skizze

Stufe I - Basisaufgabe

1. Ein Button „Start“ bzw. „Pause“ start bzw. stoppt das Spiel. Vorzugsweise wechselt ein Button die Beschriftung.
2. Am unteren Rand einer Zeichenfläche (~Spielfeld) lässt sich - mit Hilfe der Pfeiltasten - ein schraffiertes Quadrat nach links/rechts steuern. Dies ist der „Avatar“ des Spielers.
3. Ein Button „Speichern“ sichert den Zustand des Spiels und seiner Objekte in ein „savegame“.
4. Ein Button „Laden“ lädt die Informationen aus dem „savegame“ und stellt den Zustand wieder her.
5. Links oben zeigt eine Zahl einen Punkte-Counter an. Er steigt gleichmäßig und regelmäßig an.
6. Rechts oben zeigen drei ausgefüllte geometrische Formen (zB Kreise oder Herzen) die drei Leben des Spielers an.
7. Das Spiel generiert Objekte, die das Spielfeld am oberen Rand betreten und sich zum unteren Rand bewegen. Wenn Sie diesen überschreiten werden die Objekte aus dem Spiel entfernt. (ggf. Stufe II beachten!)

Stufe II

Die Basisaufgabe wird realisiert. Zusätzlich: Das Spiel generiert zufällige, unterschiedliche Objekte, die das Spielfeld am oberen Rand betreten und sich zum unteren Rand bewegen. Die unterschiedlichen Objekte werden durch verschiedene Konfigurationen einer Klasse erzeugt. Sie unterscheiden sich in den Parametern Geschwindigkeit, Form und Farbe. Gleiches Aussehen (Form & Farbe) bedeutet gleiches Verhalten (Bewegung).

Kollidiert eines der Objekte mit dem „Avatar“ des Spielers, so verliert dieser ein Leben und das Spiel „friert“ für einen Augenblick ein. Verliert der Spieler ein Leben, so wird eine Darstellung der Spielerleben nicht mehr ausgefüllt, sondern als leerer Rahmen angezeigt. Verliert er das letzte Leben Endet das Spiel, indem es kurz „einfriert“ und schließlich neu startet.

Stufe III

Stufe II + Zusätzlich: Wenn der Spieler Leben verloren hat, wird sein „Avatar“ abgeschwächt dargestellt. Eine neuer Typ von Objekten, der selten vorkommt und sich optisch deutlich abhebt regeneriert bei Kollision ein Leben, statt es zu entfernen. Das Spiel wird trotzdem kurz eingefroren. Verliert der Spieler werden „Game over“ und der Punktestand groß angezeigt.

Stufe IV

Stufe III + Zusätzlich: Eine Kategorie von Objekten, die sich von oben durch das Spielfeld bewegen „schlägt Haken“ über das Spielfeld. Es bewegt sich also im Zick-Zack. Wann immer es eine bestimme Strecke zurückgeleget hat, wird der Winkel der Bewegung verändert. Diese neuen Objekte treten seltener auf.
Die Bewegungen (gerade und Zick-Zack) sind als wiederverwendbare Funktionen anzulegen.

* Beide Aufgabenteile sind in einem git repository auf GitHub.com anzufertigen. Eingereicht wird der Archiv-Download des repositories. Die Adresse des repositories ist bei der Einreichung anzugeben. **AUFGABENTEIL 2 IST KEINE GRUPPENARBEIT.**
Es wird empfohlen zunächst die vollständige Aufgabenstellung des Aufgabenteils zu lesen, um ggf. Anforderungen späterer Stufen frühzeitig bei der Planung zu bedenken. Informationen zu Fristen und Bewertung finden sich auf der Webseite der HKI.