Semesterprojekte - Entwicklung von Multimediadiasystemen

Sebastian Bauer*

April 2, 2014

Contents

*mail@sebastianbauer.info

1 Einführung und allgemeine Ziele

Es soll während des Semesters eine Qt-Widget-Anwendung in 2er-Gruppen entwickelt werden, die eines der in diesem Dokument vorgestellten Probleme löst. Die Anwendung ist durch eine fehlerfreie Implementationen und durch einen Vortrag mit einer Diskussion der Ergebnisse am Ende des Semesters in der Laborübungsgruppe oder im Unterricht vorzustellen. Die wesentlichen Aspekte der Applikation sowie die Vorgehensweisen zur Entwicklung dieser sollen in einer ca. 10-seitigen Ausarbeitung dokumentiert werden.

1.1 Projektunabhängige Anforderungen

Im Allgemeinen werden folgende Anforderungen an die Applikation gestellt:

1. (1 point) Die Anwendung ist eine Qt-Widget-Anwendung mit einem Hauptfenster (*single document interface*).
2. (1 point) Die Größe des Fensters kann mit den üblichen Methoden verändert werden, das GUI passt sich der Größenveränderung automatisch an.
3. (1 point) Es gibt es einen Vollbild-Modus
4. (4 points) Die Anwendung ist lokalisiert und das Benutzerinterface ist in mind. zwei Sprachen (z.B. englisch und deutsch) verfügbar.
5. (4 points) Die Applikation läuft auf einem Desktop-Rechner, wird aber auch auf einer mobilen Plattform getestet und wenn nötig angepasst. Das Ergebnis ist in der Dokumentation festzuhalten.
6. (4 points) Die Anwendung regiert weiterhin auf Benutzereingaben, wenn längere Berechnungen anstehen. Beispielsweise soll es möglich sein, eine länger dauernde Berechnung abzubrechen.

Für diesen Bereich gibt es max. **15 Punkte**.

1.2 Anforderungen an den Entwicklungsprozess

An den Entwicklungsprozess werden folgende Anforderungen gestellt:

1. (2 points) Es wird ein Versionsverwaltungssystem wie *git* benutzt.
2. (4 points) Klassen, Funktionen und Schnittstellen sind ausführlich im Quelltext im Doxygen- oder qdoc-Format dokumentiert. Die Doxygen- bzw. qdoc-basierende Dokumentation lässt sich einfach mittels *make* generieren.
3. (1 point) Verwendete Algorithmen und Datenstrukturen sind möglichst QT-unabhängig zu entwickeln, um eine anderweitige Nutzung zu gewährleisten.

4. (4 points) Fehlerfreies Funktionieren der verwendete Algorithmen und Datenstrukturen ist mit Unit-Tests zu belegen und deren Erfolg ist im Beleg zu dokumentieren.

5. (4 points) Werkzeuge wie *valgrind* und *AddressSanitizer* werfen bei der Ausführung des Programms und der Unit-Tests keine Fehlermeldungen auf. Der Erfolg ist im Beleg zu dokumentieren.

Für diesen Bereich gibt es maximal **15 Punkte**.

Die Quellen können z.B. auf dem F4-Projekte-Server verwaltet werden oder auf Open-Source-Plattformen wie BitBucket, GitHub, Gitorious oder SourceForge.

1.3 Projektspezifische Anforderungen

Weitere Anforderungen sind in den einzelnen Projektbeschreibungen ausformuliert. Hier gibt es maximal **30 Punkte** zu erreichen.

1.4 Eigene Anforderungen

Weitere fünf sinnvolle Anforderungen sind von der Gruppe selbst zu formulieren. Diese sollen spätestens bis zur vierten Veranstaltungswoche (24. April) im Kurs vorgetragen und dem Dozenten per E-Mail übermittelt werden. Für diese Teilaufgabe gibt es maximal **10 Punkte**, davon 1 Punkt je Anforderungsformulierung und jeweils 1 Punkt für die erfolgreiche Umsetzung einer Anforderung. Bei Bedarf (z.B. bei komplexeren Ideen) kann auch in vorheriger Absprache mit dem Dozenten eine andere Aufteilung vorgenommen werden.

1.5 Vortrag

Der Vortrag sollte eine Dauer von ca. **25 Minuten** haben und muss im Laufe der Veranstaltung zu einem von den Projektteilnehmern frei wählbaren Termin erfolgen. Für den Vortrag gibt es keine Bewertungspunkte, er ist für das erfolgreiche Bestehen der Lehrveranstaltung jedoch notwendig.

1.6 Beleg

Der Beleg soll einen Umfang von mindestens zehn Seiten hab. Für den Beleg gibt es keine Bewertungspunkte, ein Beleg von guter Qualität ist aber notwendig für das erfolgreiche Bestehen der Lehrveranstaltung.

1.7 Bereitstellung des Codes

Zeitgleich mit der Abgabe des Belegs, ist auch der Quelltext bereitzustellen. Dieses kann in Form eines Git-Repositorys oder eines Links, hinter dem sich das Repostory verbirgt, geschehen.

1.8 Bewertung

Die Bewertungspunkte für die Erfüllung einer Anforderung befindet sich neben der Anforderung. Insgesamt sind 70 Bewertungspunkte erreichbar, wobei eine Bewertung von 100% bereits bei 65 erreichten Punkten vorliegt, d.h.,

$$\text{Bewertung in Prozent} = \frac{\text{erreichte Bewertungspunkte}}{65} \cdot 100\,\%$$

Für die Note kommt die HTW-Tabelle zur Anwendung. Voraussetzung für das Bestehen ist zudem noch das Halten eines Vortrags sowie die Abgabe des Belegs.

1.9 Wichtige Termine

- Vorstellung selbst gewählter Anforderungen: 24. April 2014
- Zwischenstand: Anfang Juni 2014
- Vorträge: Ab Mitte Juni 2014
- Abgabe Programm und Beleg: Ende Juli 2014

1.10 Interesse an ein Projekt anmelden

Im Wiki zur Veranstaltung `https://studi.f4.htw-berlin.de/redmine/projects/0-mm2014/wiki` befindet sich eine Liste, in der man sein Interesse für Themen bekunden kann. Dabei gilt: Wer zuerst kommt mahlt zuerst. Bitte interessiert Euch für mindestens zwei dieser Projekte (oder wählt in Absprache mit dem Dozenten das Freestyle-Projekt), da im Kurs jedes Projekt von mindestens zwei Gruppen bearbeitet werden soll.

2 Projektbeschreibungen

2.1 Vier Gewinnt

Ziel dieses Projektes wird es sein, das Spiel *Vier gewinnt* zu implementieren. Die speziellen Anforderungen für dieses Projekt lauten:

1. (1 point) Es können zwei Menschen gegeneinander oder ein Mensch gegen den Computer spielen.
2. (1 point) Beim Spiel Mensch gegen Computer kann ausgewählt werden, wer beginnt.
3. (1 point) Menschliche Spieler sollen vor Spielbeginn in der GUI ihre(n) Namen eingeben
4. (2 points) Für das Spielfeld und die Spielsteine soll es mindestens drei verschiedene Designs geben, die sich nicht nur in den Farben unterscheiden.

5. (1 point) Die Anzahl der Zeilen und Spalten des Spielfeldes soll vor Spielstart in der GUI konfigurierbar sein (z.B. 5 bis 9 Zeilen und Spalten, voreingestellt 7x6 wie im echten Spiel)

6. (1 point) Die Spielsteine sollen hinter dem Spielfeld durch die Löcher in ihre Endposition fallen, wobei das Fallen der Spielsteine animiert ist und z.B. durch leichtes Bouncing am Ende abgeschlossen wird.

7. (1 point) Beim Fallen/Aufprall soll ein Sound abgespielt werden, für jede Partei ein anderer.

8. (1 point) Die Anwendung lässt illegale Züge nicht zu und weist sie mit Animation und Sound zurück.

9. (1 point) Die Anwendung erkennt eine Gewinnposition und ein Unentschieden, spielt dazu geeignete Sounds ab und zeigt das Resultat in einer Animation an.

10. (1 point) Alle Spielergebnisse sollen in einer lokalen Datenbank (SQLite) abgespeichert werden und alle wichtigen Informationen über die gespielten Spiele enthalten.

11. (1 point) Die Spielergebnisse sollen in der GUI abrufbar sein (Highscore)

12. (1 point) Die Spielergebnisse (Highscore) sollen ausgedruckt werden können.

13. (1 point) Es soll möglich sein, ein Spiel zu unterbrechen (Save) und später, nach dem Neustart der Anwendung fortzusetzen (Load).

14. (1 point) Die Anwendung soll es Spielern erlauben, über das Netzwerk zu spielen. Es darf davon ausgegangen werden, dass die beiden Anwendungen im selben Subnetz sind, und die Ports erreichbar sind (keine Firewall, keine NAT, etc.)

15. (1 point) Die Anwendung soll sowohl Remote-Verbindungen annehmen können (als Server agieren), als auch sich mit einer anderen Anwendung aktiv verbinden können (als Client agieren).

16. (1 point) Eintreffende Verbindungwünsche sollen angezeigt werden und akustisch hinterlegt werden

17. (1 point) Der Anwender des Servers kann entscheiden, ob er den Verbindungswunsch annimmt, oder nicht.

18. (1 point) Der Anwender des Clients wird über die Entscheidung informiert.

19. (1 point) Die Verbindung soll solange gehalten werden, bis das Spiel beendet ist.

20. (5 points) Beim Spiel Mensch gegen Computer soll eine einfache Spielstrategie (Heuristik) implementiert werden. Es gibt eine Reihe von Arbeiten, die sich mit dem Algorithmus beschäftigen z.B. `http://www.mathematik.uni-muenchen.de/~spielth/artikel/VierGewinnt.pdf`, `http://www.ke.tu-darmstadt.de/bibtex/attachments/single/124`

21. (5 points) Beim Spiel Mensch gegen Computer soll die Spielstärke des Computers in mindestens 3 Stufen einstellbar sein

2.2 Reversi/Othello

Für dieses Projekt soll das Spiel Reversi mittels QT implementiert werden. Die speziellen Anforderungen sind ähnlich wie beim Spiel *Vier gewinnt*.

1. (1 point) Es können zwei Menschen gegeneinander oder ein Mensch gegen den Computer spielen.
2. (1 point) Beim Spiel Mensch gegen Computer kann ausgewählt werden, wer beginnt.
3. (1 point) Menschliche Spieler sollen vor Spielbeginn in der GUI ihre(n) Namen eingeben
4. (2 points) Für das Spielfeld und die Spielsteine soll es mindestens drei verschiedene Designs geben, die sich nicht nur in den Farben unterscheiden.
5. (1 point) Die Größe des Spielfelds soll vor Spielstart in der GUI konfigurierbar sein (z.B. 5 bis 10 Zeilen und Spalten, voreingestellt 8x8 wie im echten Spiel)
6. (1 point) Beim Setzen eines Spielsteines soll ein Sound abgespielt werden, für jede Partei ein anderer.
7. (1 point) Das Umdrehen eines gegnerischen Spielfelds soll animiert sein.
8. (1 point) Die Anwendung lässt illegale Züge nicht zu und weist sie mit Animation und Sound zurück.
9. (1 point) Die Anwendung erkennt eine Gewinnposition und ein Unentschieden, spielt dazu geeignete Sounds ab und zeigt das Resultat in einer Animation an.
10. (1 point) Alle Spielergebnisse sollen in einer lokalen Datenbank (SQLite) abgespeichert werden und alle wichtigen Informationen über die gespielten Spiele enthalten.
11. (1 point) Die Spielergebnisse sollen in der GUI abrufbar sein (Highscore)
12. (1 point) Die Spielergebnisse (Highscore) sollen ausgedruckt werden können.
13. (1 point) Es soll möglich sein, ein Spiel zu unterbrechen (Save) und später, nach dem Neustart der Anwendung fortzusetzen (Load).
14. (1 point) Die Anwendung soll es Spielern erlauben, über das Netzwerk zu spielen. Es darf davon ausgegangen werden, dass die beiden Anwendungen im selben Subnetz sind, und die Ports erreichbar sind (keine Firewall, keine NAT, etc.)
15. (1 point) Die Anwendung soll sowohl Remote-Verbindungen annehmen können (als Server agieren), als auch sich mit einer anderen Anwendung aktiv verbinden können (als Client agieren).

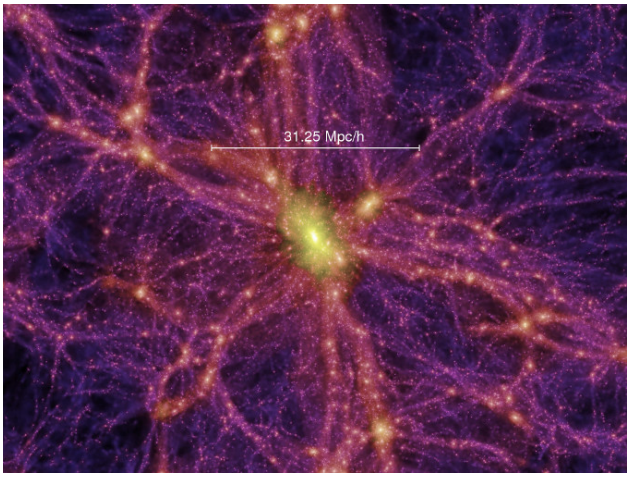

Figure 1: **N-Körper-Simulation**. Dieses Bild entstammt aus dem Millennium Ru, einer Simulation des Universums von seiner Entstehung bis heute. Quelle: Wikipedia

16. (1 point) Eintreffende Verbindungwünsche sollen angezeigt werden und akustisch hinterlegt werden

17. (1 point) Der Anwender des Servers kann entscheiden, ob er den Verbindungswunsch annimmt, oder nicht.

18. (1 point) Der Anwender des Clients wird über die Entscheidung informiert.

19. (1 point) Die Verbindung soll solange gehalten werden, bis das Spiel beendet ist.

20. (5 points) Beim Spiel Mensch gegen Computer soll eine einfache Spielstrategie (Heuristik) zum Beispiel auf Basis des Minimax-Algorithmus implementiert werden. Eine kurze Beschreibung findet sich z.B. auf `http://mnemstudio.org/game-reversi-example-2.htm`.

21. (5 points) Beim Spiel Mensch gegen Computer soll die Spielstärke des Computers in mindestens 3 Stufen einstellbar sein

2.3 N-Körper-Simulation

Das N-Körper-Problem beschäftigt sich mit der Simulation der Bewegung einer fixen Menge von N Körpern, die durch Kräfte miteinander interagieren. Dies könnten z.B. unsere Himmelskörper sein. Dabei ist für jeden Körper i eine Masse m_i, ein initialer Positionsvektor $\vec{p}_i$ sowie ein initialer Geschwindigkeitsvektor $\vec{v}_i$ gegeben.

Die Beschleunigung, die ein Körper i von den anderen Körpern im System erfährt, ergibt sich näherungsweise aus

$$\vec{a}_i = \gamma \sum_{j=1}^{N} \frac{m_j \vec{r}_{ij}}{(\|\vec{r}_{ij}\|^2 + \epsilon^2)^{\frac{3}{2}}},$$

wobei γ die Gravitationskonstante, ϵ ein *softening factor*[1] und $\vec{r}_{ij} = \vec{p}_j - \vec{p}_i$ ist. Mit Hilfe der errechneten Beschleunigung, die Körper i erfährt, lässt sich seine Position sowie Geschwindigkeit iterativ durch ein Zeitschrittverfahren aktualisieren.

Ziel dieser Aufgabe ist es, ein Programm zu erstellen, die eine N-Körper-Simulation ausführt und hierbei den Benutzer zahlreiche Interaktionsmöglichkeiten bietet.

Die Werte für die Anfangsbedingungen der Planeten finden sich zum Beispiel in `http://iau-comm4.jpl.nasa.gov/XSChap8.pdf` wieder. Werte für weitere Himmelskörper lassen sich über `http://ssd.jpl.nasa.gov/?horizons` abrufen. Sinnvoll ist ein Skript für das Telnet-Interface zu schreiben, das die Eingabe-Datei erzeugt. Der Benchmark soll für mehre, auch hohe N durchgeführt werden.

1. (3 points) Der Benutzer soll die Anfangsbedingungen der zu simulierenden Körper per GUI erfassen. Der Benutzer kann neue Körper hinzufügen, existierende Körper entfernen oder die komplette Liste löschen. Es sollten mind. 10000 Körper verwaltet werden können.

2. (1 point) Der Benutzer kann die Anfangsbedingungen in ein einfach zu parsendes Format exportieren.

3. (1 point) Der Benutzer kann die Tabelle durch den Inhalt einer Datei ergänzen. Die Funktion soll kompatiblen zum Format sein, das zum Exportieren benutzt wird.

4. (2 points) Der Benutzer soll die Datei mit Koordinaten per Drag'n'Drop in die Applikation importieren bzw. ergänzen können. Der Benutzer sollte gewarnt werden, wenn die Aktion zu einem Löschen der aktuellen Daten führt.

5. (2 points) Die Anfangsbedingungen und genereller die aktuellen Werte der Positionsvektoren sollen graphisch veranschaulicht werden können z.B. über das Qt3D Modul.

6. (1 point) Optional sollen auch die Geschwindigkeitsvektoren dargestellt werden.

7. (1 point) Per Mouse-Hover über einen Körper können die Werte des Körpers angeschaut werden.

8. (2 points) Positions- und Geschwindigkeitvektor eines Körpers kann in der graphischen Ansicht per Drag'n'Drop modifiziert werden.

9. (1 point) Der Benutzer soll direkt im Hauptfenster eingeben können (ohne separaten Dialog), welche Zeitraum die Simulation erfassen soll.

10. (1 point) Der Benutzer soll direkt im Hauptfenster eingeben können (ohne separaten Dialog), wie viele Schritte die Simulation haben soll.

11. (5 points) Die Applikation soll das N-Körper-Problem über die eingegeben Daten lösen. Der Rechenkern ist QT-unabhängig zu entwickeln (z.B. Iterator-Pattern oder Callback-Pattern).

[1] Damit wird verhindert, dass bei einer Kollision eine Division durch 0 stattfinden. Sinnvolle Werte sind z.B. $\epsilon = 0.01$.

Figure 2: **Münzenbild**.

12. (1 point) Die Simulation soll vom Benutzer über ein Button gestartet werden können.

13. (2 points) Um bei größeren Simulationen den Speicherbedarf der Applikation nicht in die Höhe schnellen zu lassen, sollen die gewonnen Daten in eine SQLite-Datenbank gespeichert werden.

14. (2 points) Während die Simulation läuft, soll die graphische Ansicht aktualisiert werden. Hierbei ist zu beachten, dass nicht unbedingt jeder Zeitschritt dargestellt wird. D.h., die Aktualisierungsfrequenz der Darstellungen muss nicht mehr als zwischen 10 bis 25 Bilder/s betragen.

15. (1 point) Der Benutzer kann die Berechnung abbrechen.

16. (2 points) Der Benutzer kann auch über eine Zeitleiste durch die bereits durch die Simulation berechneten Zeitpunkte bewegen.

17. (2 points) Der Benutzer soll Daten leicht über das Horizons-Projekt `http://ssd.jpl.nasa.gov/?horizons` direkt aus dem Programm importieren können.

2.4 Zählen von Kreisen

Gegeben ist ein zweidimensionales Bild, z.B. ein Foto von Geldmünzen wie in Abbildung 2. Es soll algorithmisch die Anzahl der Kreise bestimmt werden, die auf dem Bild zu sehen sind. Hierfür eignet sich die sogenannte Hough-Transformation, die den Bildraum in einem Parameterraum der zu suchenden Figurklasse überführt. Bei Kreisen sind dies Position und Radius.

Als Eingabe der Hough-Transformation dient häufig ein Binärbild, auf dem nur die Umrisse der Objekte zu sehen sind. Aus einem Foto kann dieses durch einen Kantendetektionsfilter wie z.B. dem Sobel-Filter gewonnen werden. Ein guter Einstiegspunkt für mehr

Information über den Algorithmus ist der Wikipedia-Artikel `http://de.wikipedia.org/wiki/Hough-Transformation`.

Neben den bereits formulierten allgemeinen Anforderungen, sind für diese Applikation folgende Anforderungen zu realisieren

1. (1 point) Der Benutzer soll ein Schwarz-Weiß Bild per Dateiauswahlfenster in die Applikation laden können.
2. (1 point) Der Benutzer soll ein Schwarz-Weiß Bild per Drag'n'Drop in die Applikation laden können.
3. (5 points) Die Applikation soll die wahrscheinlichste Anzahl der Kreise, die per Hough-Transformation im geladenen Bild detektiert worden sind, als Zahl ausgeben.
4. (1 point) Die detektierten Kreise sollen hervorgehoben werden, z.B. indem für jeden Kreis ein Kreis in einer bestimmten Farbe über das Originalbild gezeichnet wird, so wie in Abbildung 2.
5. (1 point) Ob die Kreise hervorgehoben werden, ist vom Benutzer direkt in der Benutzeroberfläche einstellbar.
6. (1 point) Die Farbe der Hervorhebung kann vom Benutzer frei gewählt werden.
7. (1 point) Die Inhalte der detektierten Kreise auf dem Originalbild können in eine Datei exportiert werden. Die Datei kann z.B. die Originalgröße besitzen und alle Pixel außerhalb der detektierten Kreise werden mit voller Transparenz gespeichert.
8. (1 point) Die Inhalte der detektierten Kreise auf dem Originalbild können in die Zwischenablage kopiert werden.
9. (2 points) Der Benutzer soll in einem Histogram-Viewmode schalten können, in dem der Parameterraum angezeigt wird.
10. (2 points) Per Mouse-Hover kann der Benutzer den Histogram-Viewmode explorieren.
11. (5 points) Der Benutzer soll ein beliebiges Bild, zum Beispiel ein Foto, in die Applikation laden können. Die Applikation soll im Falle eines Nicht-Zwei-Farbbildes, die nötigen Vorfilterschritte selbst übernehmen. Der Benutzer wird darüber in Kenntnis gesetzt.
12. (1 point) Es werden mehrere gängige Dateiformate für das Importieren unterstützt.
13. (1 point) Das Bild soll von einer Kamera eingelesen werden.
14. (2 points) Über eine Text2Speech-Lösung, soll die Anzahl der detektierten Kreise mit Hilfe von Sprache ausgegeben werden.

15. (5 points) Das Bild von der Kamera soll fortwährend prozessiert (Detektion und Hervorhebung der Kreise). Es wird eine zuschaltbare Approximation verwendet (z.B. Verringern der Auflösung des Bildes) und die Algorithmen werden parallelisiert, um eine möglichst geringere Verzögerungszeit zu erreichen. Dies ist besonders auf mobilen Plattformen interessant, so kann man z.B. Live-Bild der Mobiltelefonkamera alle Kreise hervorheben und ggfs. weiterverarbeiten (z.B. Verkehrsschilderkennung).

2.5 Freestyle

In dieser Aufgabe kann das Thema des Projektes in vorheriger Absprache mit dem Dozenten frei gewählt werden. Die projektspezifischen Anforderung dürfen hier selbst formuliert werden und sollen sich vom Umfang und Schwierigkeit an den Anforderungen der vorgegeben Projekte orientieren. Die allgemeinen Anforderungen sowie die Anforderungen an den Entwicklungsprozess bleiben bestehen.

Die Ausformulierung von Projektidee und Anforderungen muss dem Dozent bis spätestens zur vierten Veranstaltungswoche schriftlich vorliegen. Das Thema ist akzeptiert, sobald der Dozent sein Einverständnis erteilt. Der Dozent behält sich jedoch vor, die Anforderungen ggfs. anzupassen. Bei besonders interessanten Projektideen besteht die Möglichkeit, weitere **5 Bonuspunkte** zu erhalten, so dass man maximal 75 Punkte erreichen kann.