Programmentwurf OOP in C++: KITTI Reaction Game

DHBW Stuttgart, 2024
Christian Bader, Christian Holz

I. Zusammenfassung

Entwickeln Sie ein Reaktionsspiel basierend auf dem KITTI-Datensatz, bei dem Bilder des Datensatzes mit dazu passenden Ground Truth (GT) Bounding Boxen visualisiert werden. Die Aufgabe des Spielers ist es, möglichst schnell eine der Boxen anzuklicken, welche je nach Spielmodus definiert wird.

II. Details: Funktionale Anforderungen

Datensatz: Der KITTI Datensatz ist ein Datensatz für die Entwicklung von Umgebungserfassung für autonomes/automatisiertes Fahren. Laden Sie die „Left color images of tracking data set", sowie die "Training labels of tracking data set" von der Website herunter (The KITTI Vision Benchmark Suite (cvlibs.net)). Somit erhalten Sie die Bilder eines Datensatzes, sowie die dazugehörenden GT Bounding Boxen, welche in Sequenzen unterteilt sind.

Benutzeroberfläche (GUI): Entwickeln Sie eine Menüführung, welches zu Beginn des Spiels über die Konsole den Namen des Spielers, die Anzahl N der zu spielenden Bilder, die Sequenz aus der diese Bilder stammen sollen, sowie den Spielmodus einliest. Entwickeln Sie ein Fenster, das anschließend die Bilder anzeigt und die GT Boxen einblenden kann. Dabei sollen alle Boxen vom Typ „DontCare" ignoriert werden. Falls ein Bild keine passenden GT Boxen enthält, soll es übersprungen werden.

Spielmechanik: Zeigen Sie das erste Bild der ausgewählten Sequenz. Nach dem Anzeigen des Bildes wird eine Reaktion vom Spieler erwartet. Je nach Spielmodus muss sich der Spieler unterschiedlich verhalten.

Nachdem der Spieler reagiert hat, wird das nächste Bild angezeigt, bis N Bilder gespielt wurden. Sollte der Spieler nicht oder zu langsam reagieren, sollte nach einer definierten Zeit (3 Sekunden) das nächste Bild geladen werden und eine Info in der Konsole erscheinen. Sollte der Spieler einen falschen Bereich auf dem Bild anklicken, sollte ebenfalls eine Info an den Spieler ausgegeben und eine definierte Strafzeit (5 Sekunden) als Reaktionszeit gespeichert werden.

Die Spielmodi sind auf der folgenden Seite definiert.

Mode 1 - Direct Click Reaction: Zeichnen Sie genau eine (zufällig ausgewählte) GT Box in Rot in das Bild. Der Spieler muss so schnell wie möglich auf diese rote Box klicken, wobei die Reaktionszeit nach anzeigen des Bildes gemessen wird

Abbildung 1: Spielfenster - Direct Click Reaction mode

Mode 2 - Color Change Reaction: Zeichnen Sie alle GT Bounding Boxen in der einheitlichen Farbe Blau in das Bild. Nach einer zufälligen Zeit (zwischen 1-2 Sekunden) ändert sich die Farbe einer zufälligen Box zu Rot. Der Spieler muss so schnell wie möglich auf die rote Box klicken. Anschließend muss die Auswahl mit der Leertaste bestätigt werden. Die Reaktionszeit bis zum Betätigen der Leertaste nach dem Wechsel der Farbe wird gemessen. Bei falscher ausgewählter Box wird die Strafzeit angenommen.

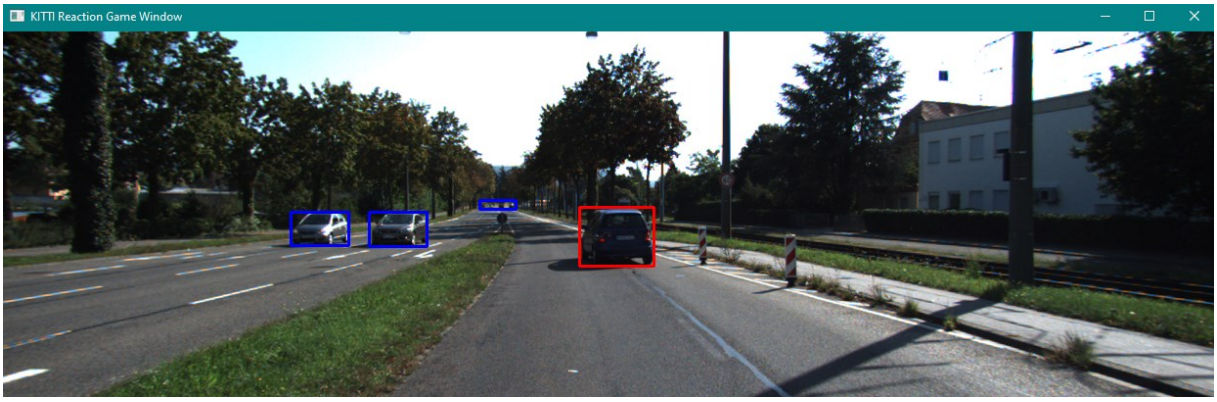

Abbildung 2: Spielfenster - Color Change Reaction mode

Zeitmessung: Speichern Sie die Reaktionszeit jedes Bildes für die Ausgabe der Ergebnisse nach Beendigung des Spiels.

Feedback und Ergebnisse: Geben Sie dem Spieler am Ende des Spiels Feedback über seine Reaktionszeit, wobei der eingelesene Name angezeigt wird. Zeigen Sie am Ende des Spiels eine Übersicht der durchschnittlichen Reaktionszeit und die besten drei Ergebnisse mit Bildnummer an.

Weitere Anforderungen: Das Programm muss über eine main.cpp verfügen und in der Konsole starten können. Überprüfen Sie, ob die Rückgabewerte des Anwenders bei der Menüführung in der Konsole sinnvoll sind und reagieren Sie entsprechend.

III. Details: Formale Anforderungen Code Abgabe

Prüfungsleistung: Die Programmieraufgabe stellt eine Prüfungsleistung dar, die eigenständig (bzw. von den Teilnehmern eines 2er/3er Teams) zu erbringen ist. Jede Übernahme oder Weitergabe von Codefragmenten von/an anderen Teilnehmer stellt einen Täuschungsversuch dar.

Klassendiagramm: Schreiben Sie ein Klassendiagramm in UML, welches die wichtigsten Klassen und die wichtigsten Public-Methoden enthält (keine privaten Methoden, keine Tests etc.).

Code-Kommentare: Kommentieren Sie Ihren Code, um die Lesbarkeit zu erhöhen.

Code-Formatierung: Das Programm soll sinnvoll formatiert sein (vertikale und horizontale Einrückung, Struktur, siehe Coding Conventions).

Clean-Code: Schreiben Sie Clean-Code nach allen Regeln der Kunst!

Unit-Tests: Schreiben Sie Unit-Tests für Ihr Programm. Mindestanforderung ist, dass alle Klassen mindestens einen sinnvollen Unit-Test haben. Zudem sollen alle relevanten, logikbeinhaltenden Methoden getestet werden.

GitHub Repository: Ihr Projekt muss ein eigenes GitHub Repository (pro Team) sein, welches Ihr Dozent selbst zum Abgabezeitraum klonen kann. Schalten Sie mich hierfür Ihren Dozenten frei: Kurs ITA: ncPtKLKN, Kurs IN: PandaLehre

README.md: Schreiben Sie eine README.md Datei, in der Sie (auf nicht mehr als etwa zwei Seiten) Ihre Implementierung beschreiben, insbesondere alle Erweiterungen, die Sie vorgenommen haben.

Die README muss ebenfalls eine Anleitung enthalten, was ausgeführt werden muss, um Ihr Programm zu starten (z.B. Einbinden von GoogleTests etc., Kompilieren). Das Programm muss ohne Fehler auf einem Windows-PC kompilieren und starten (Sie sind jedoch nicht für externe Libraries verantwortlich).

Abgabetermin: Abgabetermin ist der 17.07.2024 um 23:59. Der letzte Push vor diesem Termin wird gewertet.

IV. Details: Formale Anforderungen Projektvorstellung

1. Kurzvorstellung der Highlights: 5 Minuten
   (eventuelle Erweiterungen, besonders schön erstellte Klasse/Struktur/UML)
2. Live-testing mit Fragen: 10 Minuten
   Das Programm muss auf Ihrem Rechner lauffähig sein. Zeigen Sie die Funktionalität beantworten Sie Fragen zu möglichen Fehlern und Implementierungen.

V. Erweiterungen

Sie können die Note durch Erweiterungen verbessern. Hier einige Vorschläge:

- Erstellen Sie einen weiteren Spielmodus, wie beispielsweise Reaktion auf Änderung der Form GT Box zu einem Kreis oder die Aufgabe ein Objekt vom Typ „Pedestrian“ anzuklicken.
- Fügen Sie ein Scoreboard ein, welches die Ergebnisse mit Namen in einem externen File abspeichert.
- … wenn Ihnen noch sinnvolle Erweiterungen einfallen, dann los …

VI. Bewertungskriterien

Insgesamt werden für die Erfüllung der Projektaufgabe 100 Punkte vergeben. Zusätzlich gibt es bis zu 10 Bonuspunkte für nicht direkt geforderte, aber sinnhafte Erweiterungen. Es ist sehr zu empfehlen, erst einmal die Kernaufgabe zu lösen!

In der Bewertung wird zwischen harten und weichen Kriterien unterschieden. Jedem Kriterium wird eine Punktzahl zugeordnet. Mindestvoraussetzung zum Bestehen sind insgesamt 50 Punkte sowie mindestens 50% der Punkte der harten Kriterien.

Punkte	Kriterium
30	Das Programm erfüllt die geforderten funktionalen Anforderungen
20	Das Programm ist objektorientiert und verwendet C++ und objektorientierte Prinzipien, wie etwa Klassen, Vererbung und Polymorphie, const-Correctness, STL-Nutzung sinnvoll und korrekt.
10	Projektbeschreibung und Klassendiagramm wie gefordert vorhanden
15	Das Programm ist einfach lesbar, strukturiert und weitestgehend Code-Smell frei (siehe *Clean Code*). Ausnahmen werden im Zweifelsfall begründet.
10	Alle relevanten, logikbeinhaltenden Methoden sind mit Unit-Tests abgedeckt
5	Korrektes Fehlerhandling (was passiert z.B. bei einer unerwarteten Eingabe?)
10	Projektvorstellung (zeitliche Vorgabe eingehalten, Q&A Session)
10	*Bonuspunkte für Erweiterungen*

Viel Erfolg!