Programmentwurf OOP in C++: Head Reaction Game

DHBW Stuttgart, 2025
Christian Bader, Christian Holz

I. Zusammenfassung

Entwickeln Sie ein Reaktionsspiel basierend auf einem Gesichtsdetektionsansatz, bei dem die Kamera ihres Laptops für verwendet wird. Die Aufgabe des Spielers ist es, herabfallenden Objekten mit dem Gesicht auszuweichen oder diese einzufangen.

II. Details: Funktionale Anforderungen

Gesichtserfassung: Verwenden Sie einen Ansatz zur Gesichtsdetektion aus einer 3rd-Party Bibliothek, wie beispielsweise den Haar Cascade Detektor oder den DNN-based Detektor von OpenCV. Dabei soll die Position und die Größe erfasst und im Kamerabild als Rechteck dargestellt werden. Falls für einzelne Bilder kein Gesicht erkannt wird, soll das Ergebnis des vorherigen Zeitschrittes oder ein Standardwert verwendet werden.

Benutzeroberfläche (GUI): Entwickeln Sie eine Menüführung, welches zu Beginn des Spiels über die Konsole den Namen des Spielers, sowie den Spielmodus einliest. Entwickeln Sie ein Fenster, das anschließend die Kamerabilder, sowie das Ergebnis des Gesichtsdetektors und der eingeblendeten Objekte anzeigt. Zum Ende des Spiels soll ein Game-Over Schriftzug angezeigt werden und eine Möglichkeit zum schließen des Fensters (zum Beispiel über den Escape Button) gegeben werden. Nachdem das Fenster geschlossen wurde, soll der Score angezeigt werden.

Spielmechanik: Im Kamerabild fallen verschiedene Geometrische Formen herunter. Diese interagieren mit dem Rechteck des Gesichtsdetektors, sodass der Spieler dafür zum Beispiel Punkte erhält, verliert oder das Spiel beendet wird. Die Formen sollen verschiedene Farben annehmen können und verschiedene, zufällige Größen und Fallgeschwindigkeiten innerhalb von selbst definierten Grenzen erhalten. Der Score soll zu jedem Zeitpunkt im Kamerabild angezeigt werden.

Die Spielmodi sind auf der folgenden Seite definiert.

Mode 1 – Dodge the balls: Der Spieler soll herabfallenden Kreisen verschiedener Größe und Geschwindigkeit mit einer zufälligen Farbe aus Rot, Blau und Grün ausweichen. Für jeden ausgewichenen Kreis (der den Boden des Kamerabilds erreicht) erhält der Spieler einen Punkt. Sobald er einen der Bälle mit dem Rechteck des Gesichtsdetektors berührt, ist das Spiel vorbei.

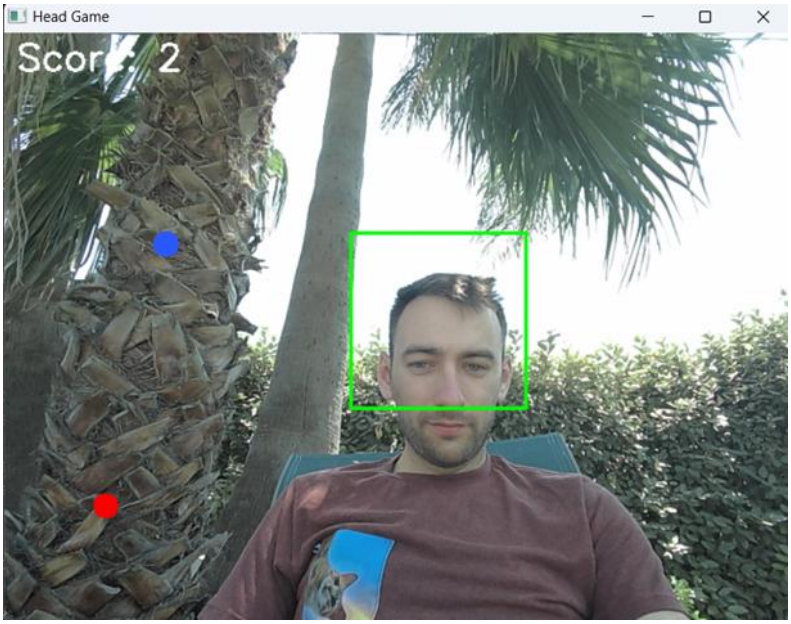

Abbildung 1: Spielfenster – Dodge balls game mode

Mode 2 – Catch the Squares: Der Spieler soll Kreisen in roter Farbe mit verschiedenen Größen und Geschwindigkeiten ausweichen und grüne Quadrate mit verschiedenen Größen und Geschwindigkeiten einsammeln. Für jedes eingesammelte Quadrat erhält der Spieler einen Punkt, für jeden getroffenen Kreis verliert der Spieler einen Punkt. Er kann nie weniger als 0 Punkte haben. In einem Spieldurchgang werden N Objekte gespielt, wobei der Spieler vor Beginn des Spiels die Anzahl N über die Konsole eingeben soll.

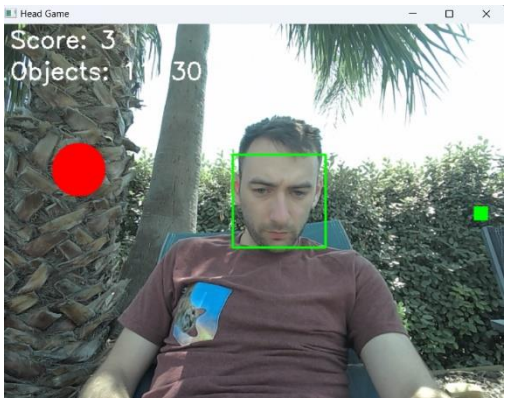

Abbildung 2: Spielfenster – Catch the squares game mode

Feedback und Ergebnisse: Geben Sie dem Spieler am Ende des Spiels Feedback über seine erspielten Punkte, wobei der eingelesene Name angezeigt wird.

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 an sinnvollen Stellen.

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 Ihre Dozenten selbst zum Abgabezeitraum klonen können. Schalten Sie uns hierfür frei: ncPtKLKN & 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 18.07.2024 um 23:59. Der letzte Push vor diesem Termin wird gewertet.

IV. Erweiterungen

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

- Erstellen Sie einen weiteren Spielmodus, bei dem beispielsweise alle Objekte vor dem Fallen auf den Boden eingesammelt werden müssen.
- Fügen Sie ein Scoreboard ein, welches die Ergebnisse mit Namen in einem externen File abspeichert.
- Erstellen Sie die Menüführung in einer GUI
- … wenn Ihnen noch sinnvolle Erweiterungen einfallen, dann los …

V. 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 32 Punkte der harten Kriterien.

Punkte	Kriterium
20	Das Programm erfüllt die geforderten funktionalen Anforderungen
35	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
20	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	*Bonuspunkte für Erweiterungen*

Viel Erfolg!