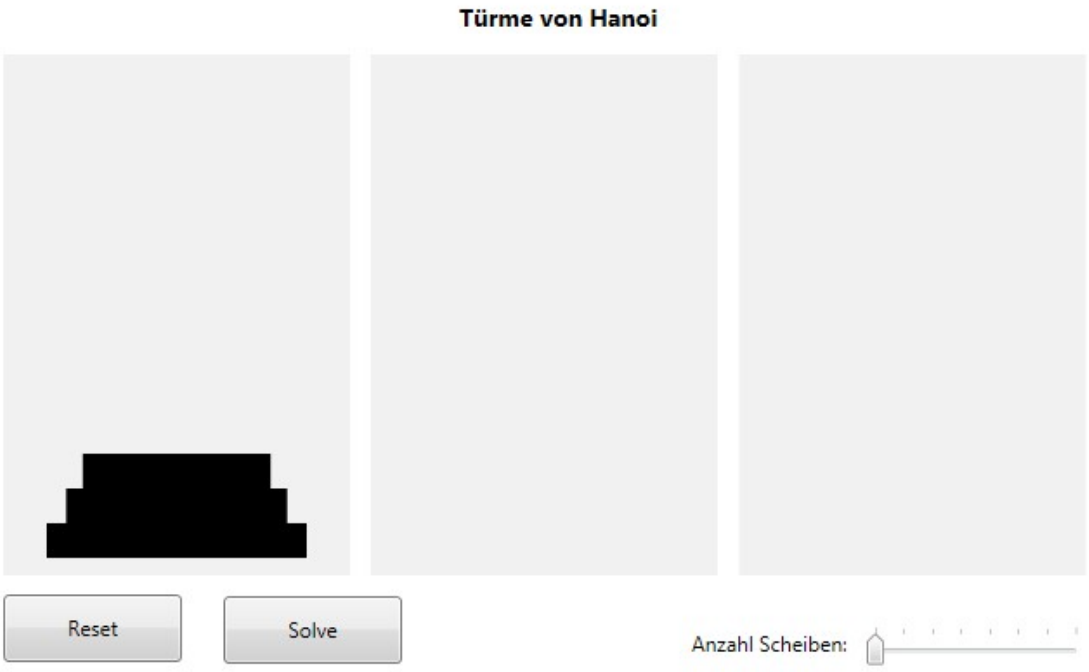
Dokumentation Hanoi

Von Duk Anh Trinh und Stephanie Schöne



1. Aufbau der Benutzeroberfläche
2. Ziel des Spiels
3. Spielmodi
4. Programmcode

1. Aufbau der Benutzeroberfläche

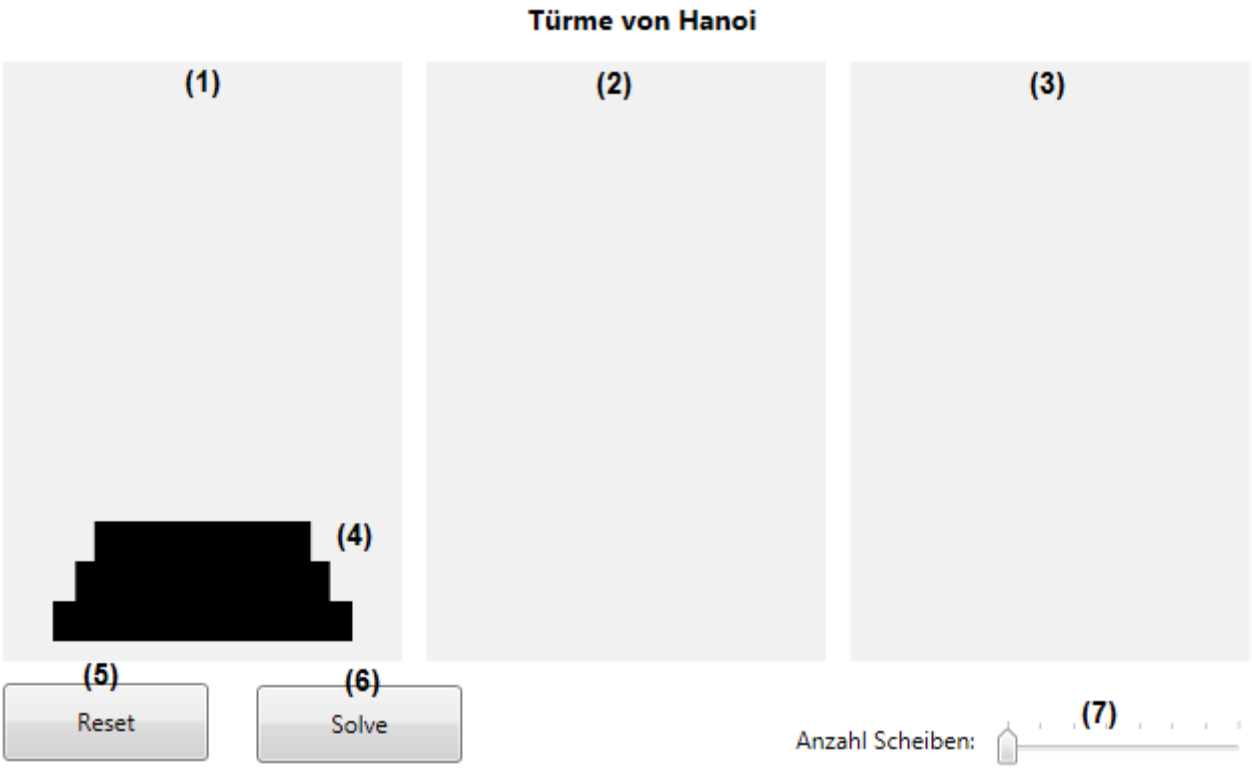


Abbildung 1: Ansicht auf die Benutzeroberfläche des Programms "Hanoi" mit nummerierten Programmelementen von (1) bis (7)

Folgend werdend die Elemente der Benutzeroberfläche, wie in Abbildung 1 nummeriert, beschrieben.

Nummer	Funktionsweise
(1)	Start Canvas: Alle Scheiben werden dort zu Spielbeginn gestapelt.
(2)	Mittlerer Canvas: Diese Fläche dient zur Zwischenlagerung von Scheiben.
(3)	Ziel Canvas: Hierhin müssen die Scheiben der Größe nach (groß unten, klein oben) sortiert werden.
(4)	Scheiben: Die schwarzen, langen Rechtecke sind die Scheiben.
(5)	Reset Button: Über diesen Button kann das Spiel zurückgesetzt werden.
(6)	Solve Button: Dieser Button erlaubt das automatische Lösen des Spieles ausgehend von der Ausgangsposition.
(7)	Schieberegler: Mit diesem Regler kann die Anzahl von Scheiben eingestellt werden. Das Minimum beträgt 3 Scheiben.

2. Ziel des Spiels

Ziel des Spiels ist es alle Scheiben durch Interaktionen von der linken Fläche (Start Canvas) zur rechten Fläche (Ziel Canvas) zu bewegen. Abbildung 2 stellt das Resultat dar. Dabei dürfen nie größere Scheiben auf kleinere gelegt werden.

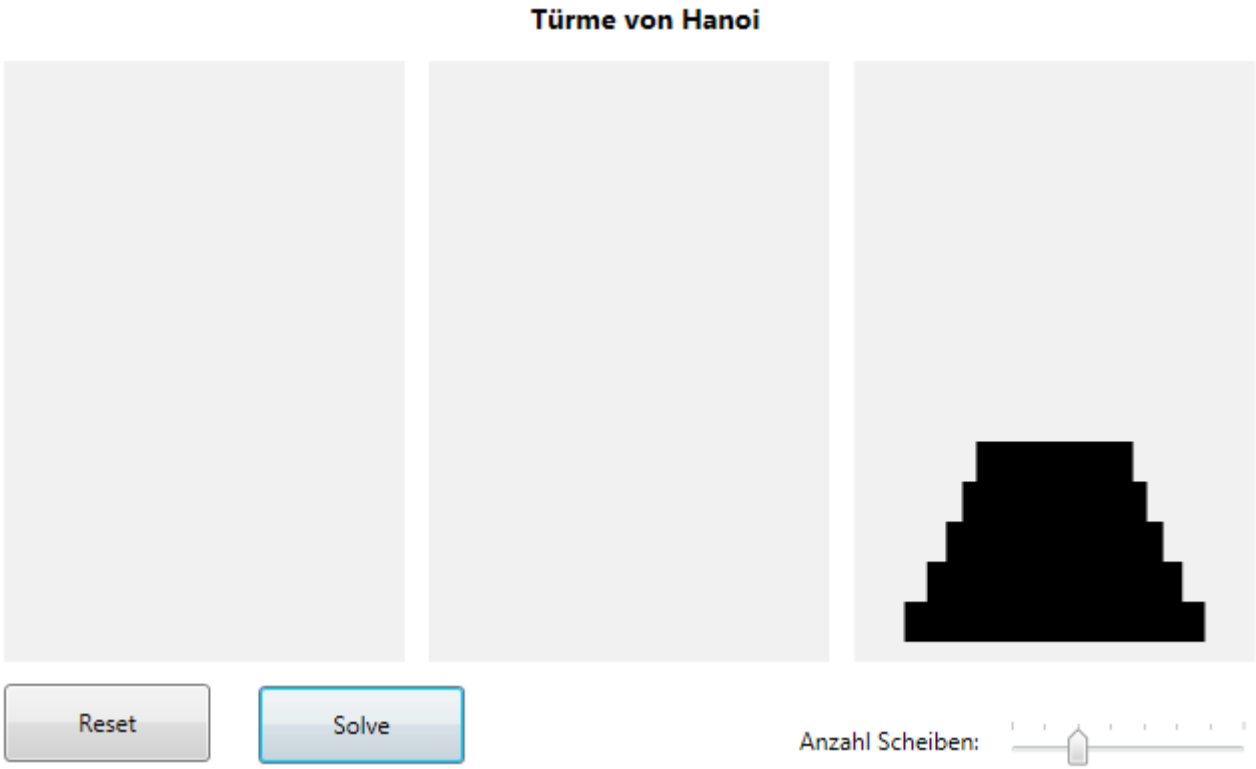


Abbildung 2: Ansicht auf die erfolgreiche Lösung des Spieles Hanoi. Alle Scheiben befinden sich in absteigender Größe im rechten Canvas.

3. Spielmodi

3.1 Manuelles Spielen

Das Spiel kann durch Mausklicks manuell gespielt werden. Dabei muss für jeden Zug ein Canvas, von dem die obere Scheibe genommen werden soll, angeklickt werden. Nur Flächen mit Scheiben werden für diese Auswahl angenommen.

Wurde eine gültige Auswahl getroffen, muss für die Verschiebung ein weiterer Canvas angeklickt werden. Hier gilt: Größere Scheiben dürfen nicht auf Kleinere platziert werden. Die Fehlermeldung „Kein gültiger Zug!“ tritt auf, wenn es dennoch versucht wird.

Wurden alle Scheiben der Größe nach (unten groß, oben klein) von **(1)** nach **(3)** sortiert, ist das Spiel gewonnen. Die Nachricht „Juhu, du hast gewonnen!“ wird angezeigt.

Durch den Reset Button **(5)** kann da Spiel jederzeit von vorn begonnen werden. Das Verändern der Scheibenanzahl **(7)** bewirkt ein automatisches Zurücksetzen.

Das Automatische Lösen funktioniert nur im Ausgangszustand.

3.2 Automatisches Spielen

Befindet sich das Spiel im Ausgangszustand, kann durch das Anklicken des Solve Button **(6)** das Spiel automatisch gelöst werden. Falls es nicht im Ausgangszustand ist, erscheint nur die Nachricht: „Reset the Game first!“.

Da für diesen Modus ein paralleler Thread verwendet wird, ist die Benutzeroberfläche noch immer verwendbar. Durch den Reset Button **(5)** kann das automatische Lösen jederzeit abgebrochen werden. Bei Veränderung der Scheibenanzahl **(7)** wird das Spiel ebenfalls zurückgesetzt.

Wird während des Vorgangs des automatischen Lösens der Solve Button **(6)** erneut angeklickt, erscheint eine Nachricht: „Reset the Game first!“. Der Vorgang wird nicht unterbrochen.

Ist der automatische Lösevorgang beendet, erscheint die Nachricht „Juhu, du hast gewonnen!“. Von hier aus muss für ein erneutes Spielen das Spiel wieder zurückgesetzt werden.

4. Programmcode

Das Programm teilt sich in die Benutzeroberfläche MainWindow.xaml und die Klasse MainWindow in MainWindow.xaml.cs auf. Die Benutzeroberfläche wurde in den vorhergehenden Abschnitten bereits beschrieben.

4.1. Globale Variablen

In der Region ***Global Parameters*** befinden sich die für das Spiel verwendeten globalen Variablen. Sie sind alle privat, da sie nur innerhalb der Klasse MainWindow verwendet werden.

Variablen	**Funktionsweise**
double canvasBottomMargin	canvasBottomMargin bestimmt den Abstand der Scheiben vom unteren Canvasrand.
double rectangleHeight	rectangleHeight bestimmt die Höhe der Scheiben.
double rectangleStartWidth	rectangleStartWidth bestimmt die Breite der ersten Scheibe.
int discAmount	siscAmount bestimmt die verwendete Anzahl von Scheiben.
System.Threading.Tasks.Task t	t baut einen parallelen Thread für das automatische Lösen auf.
CancellationTokenSource ts	Sowohl ts als auch ct werden für den Abbruch des Threads t verwendet.
CancellationToken ct	
Canvas startCanvas	In startCanvas wird der erste angeklickte Canvas für den Zug zwischengespeichert.

4.2. Initialisierung

In der Region ***Initialization*** befinden sich der Constructor MainWindow() sowie die Funktion *InitializeGame().*

Die Funktionen *InitializeComponent()* und *InitializeGame()* werden im Constructor MainWindow() ausgeführt.

InitializeComponent() initialisiert die Benutzeroberfläche in der MainWindow.xaml Datei.

InitializeGame() legt die Startwerde von canvasBottomMargin, rectangleHeight, rectangleStartWidth und discAmount fest. Anschließend setzt es das Spiel einmal durch *resetGame()* zurück, was einen Neustart erzwingt.

4.3. Spielablauf

In der Region ***Game Handling*** wird der Ablauf des Spieles kontrolliert.

Die Funktion *resetGame()* erzwingt immer den Neustart des Spieles. Sie wird durch die Veränderung des Schiebereglers **(7)** sowie den Reset Button **(5)** ausgelöst.
Zunächst kontrolliert, ob Thread t gerade das Spiel automatisch löst. Ist dies der Fall, wird der Thread durch *ts.Cancel()* abgebrochen. Die Threadabhängigen Variablen werden zurückgesetzt. Danach werden die drei Canvase, die globale Variable startCanvas zurückgesetzt, und die Funktion *startGame()* aufgerufen.

s*tartGame()* füllt den linken Canvas iterativ mit Scheiben in der durch discAmount festgelegten Anzahl. Die Iteration beginnt mit der größten Scheibe am unteren Rand des Start Canvas **(1)**.
Pro Iteration wird die vertikale Position verringert und die horizontale erhöht, damit die nächste Scheibe zentriert über die vorherige gesetzt wird. Die Breite der Scheiben sinkt mit jedem Durchlauf. Die Scheiben werden im Canvas in Form von Rechtecken, dies entspricht der internen Form Rectangle, hinterlegt.

Nach Spielstart, kann das Spiel wie in Abschnitt 3 erläutert manuell, oder automatisch gelöst werden.Für das Manuelle Lösen wird nach Bestimmung von Start und Ziel Canvas die *moveDisc()* Funktion ausgeführt.

modvDisc(Canvas targetCanvas) benutzt den globalen Parameter startCanvas und den erhaltenen Parameter targetCanvas für das Verschieben. Falls startCanvas Scheiben enthält, wird die oberste Scheibe startRect darin über *startCanvas.Children[(startCanvas.Children.Count – 1)]* mit Cast nach Rectangle erhalten.
Enthält der targetCanvas bereits Scheiben, erfolgt eine Abfrage nach wiederum dessen oberster Scheibe. Sind targetCanvas leer oder die oberste Scheibe darin größer als die des startCanvas, erfolgt die Verschiebung. Andernfalls erscheint die Nachricht „Kein gültiger Zug!“ in einer MessageBox.
Für die Verschiebung wird zunächst startRect aus dem startCanvas entfernt. Danach targetCanvas durch die *Add()* Funktion beigefügt. Eine Prüfung erfolgt, ob das Spiel nach diesem Zug gewonnen wurde. In dem Fall erscheint eine MessageBox mit der Nachricht „Juhu, du hast gewonnen!“.

Wurde Solve Button im Ausgangszustand des Spieles angeklickt, erfolgt das Erstellen des neuen Threads. ts und ct werden initialisiert, um den Thread bei Bedarf abzubrechen. Der Thread ruft die Funktion *solveAlgorithm(discAmount, LeftCanvas, RightCanvas, MidCanvas)* zur Lösung des Spiels auf.

Das automatische Lösen findet in der Region ***Automated Game Handling*** statt.
solveAlgorithm(double n, Canvas strtCanvas, Canvas tarCanvas, Canvas othrCanvas) erhält die globalen Variablen, um das Spiel rekursiv zu lösen. Die Variable discAmount bestimmt dabei die Anzahl der noch zu spielenden Züge. Für n == 1, wird eine letzte Verschiebung vom erhaltenen

strCanvas zum tarCanvas ausgeführt.
Ist n > 1 erfolgt ein rekursiver Aufruf *solveAlgorithm(n - 1, strtCanvas, othrCanvas, tarCanvas).* Diesem folgt die Verschiebeoperation und ein weiterer rekursiver Aufruf *solveAlgorithm(n - 1, othrCanvas, tarCanvas, strtCanvas).*
*V*erschiebeoperationen in dieser Funktion finden innerhalb der *this.Dispatcher.Invoke* Funktion statt. Sie aktualisiert den Hauptthread mit den in ihr ausgeführten Operationen, beginnend mit dem Festlegen der globalen Variable startCanvas = strtCanvas. Dem folgt die Funktion *moveCanvas(tarCanvas).*
Erfolgt ein Abbruchbefehl durch *ts.Cancel()* oder ist bereits eine Rücksetzung geschehen, wird die Funktion abgebrochen.

Multimodale Konzeption

Benötigte Befehle

Das Spiel sollte mit einer festen Menge an Befehlen zu steuern sein:

- Stelle Anzahl der Scheiben ein
- Starte Spiel
- Setze Spiel zurück
- Löse automatisch
- Scheibe verschieben(Start, Ziel)

Per Sprache

Die Speech-Bibliothek in .NET ermöglicht die Implementierung einer Sprachsteuerung für die Hanoi Anwendung. Dafür wird ein Auswahl von möglichen Spracheingaben benötigt. Bei korrekten Eingaben erfolgt die Ausführung der ihnen hinterlegten Logik.
Spracheingaben sollten gut verständlich und differenzierbar sein, um Eindeutigkeit zu gewährleisten.

Folgende Spracheingaben können für die Hanoi Anwendung verwendet werden:

- Anzahl Scheiben einstellen: “Anzahl Scheiben” - “Vier”
- Starte Spiel: “Start”
- Setze Spiel zurück: “Neustart”, “Reset”
- Löse automatisch: “Löse automatisch”, “Lösen”
- Scheibe verschieben(Start, Ziel): “Schiebe 1 auf 3”, “1 auf 3”

Per Gesten

Auch bei der Steuerung mit Gesten müssen die Gesten leicht aufnehmbar und unterscheidbar sein. Die Gesten sollten unabhängig von der Art der Aufnahme (Kamera, Wearables, ...) funktionieren.

Mögliche allgemeine Gesten könnten sein:

- Anzahl Schreiben einstellen: “Zeigen der Zahl mit Hilfe von Fingern”, oder: “Arm
- aufwärts oder abwärts”
- Starte Spiel: “Formen eines Dreiecks durch Bewegung einer Hand”
- Starte Spiel neu: “Formen eines Kreises durch Bewegung einer Hand”
- Löse Automatisch: “Formen eines Kreuzes mit beiden Armen”
- Scheibe verschieben(Start, Ziel): “Pack-/Greifbewegung” - “Ziehen” - “Loslass-Bewegung”

Die Hanoi Anwendung löst die Gestensteuerung mit Maus-Drag-and-Drop-Eingaben. Mit dem DTW (Dynamic Time Warp) Algorithmus werden ausgeführte Mausgesten mit hinterlegten Referenzen verglichen.

Folgende Mausbewegungen können für die Hanoi Anwendung in Frage kommen:

- Anzahl Schreiben einstellen:
  - Zeichnen einer Zahl (eventuell nicht eindeutig genug)
  - horizontalen Striches für die Richtung (mehr, weniger) entsprechend des Schiebereglers (außerhalb eines Canvas)
- (Halb)Kreis für Scheiben einstellen, danach
  - Zeichnen einer Zahl
  - vertikalen Striche: Pro Strich eine Scheibe mehr
  - horizontaler Strich für die Richtung (mehr, weniger) entsprechend des Schiebereglers
- Starte Spiel:
  - Zeichnen eines Dreiecks entsprechend eines Play-Buttons

    - Haken oder ähnlicher Winkels (auf Startcanvas)
- Setze Spiel zurück:
    - vertikaler Strich (wenn nicht bereits belegt)
    - Zeichnen eines “Z” für zurück
- Löse Automatisch: Zeichnen eines “L” oder eines Winkels, der diesem ähnelt
- Scheibe verschieben(Start, Ziel): Anklicken (und geklickt halten) des Startcanvas, Ziehen der Maus zum Zielcanvas, dann Maus loslassen
-

Angegebene Optionen müssten auf ihre Verwendbarkeit während des Spielens geprüft werden.