

C++ Kochshow

Asteroids

Detlef Wilkening

<http://www.wilkening-online.de>

22.1.2025

Wir "kochen" uns ein kleines Programm

- In C++, und "live" und in Farbe

Schauen wir erstmal, was wir heute „kochen“ wollen...

Asteroids Demo

Leider ist das in reinem ISO C++ nicht umsetzbar

- **Darum nutzen wir eine zusätzliche Bibliothek**
- **Ich habe mich für “Cinder” entschieden**
 - <http://libcinder.org/>
 - Cinder is a community-developed, free and open source library for professional-quality creative coding in C++ for Mac OS X, iOS and Windows.
 - Die aktuelle Version ist V. 0.9.2 for Windows 2015 vom 13.4.2020
 - Ich habe sie angepasst für das Microsoft Visual Studio 2022 und C++2x (latest)
 - Siehe nächste Folie
 - Die Dev-Version auf GitHub ist aktueller, was die Compiler angeht
 - <https://github.com/cinder/Cinder>
 - Ich wollte mich aber auf ein offizielles Release abstützen
 - Habe den Code vor langer Zeit auch mal unter Mac OS X mit XCode getestet

Änderungen für MSVS 2022 und C++2x (latest) 1/5

- Datei: include\cinder\app\AppBase.h
- Zeile 410/411

```
template<typename T>  
typename std::result_of<T()>::type dispatchSync( T fn );
```

=>

```
template<typename T>  
typename std::invoke_result<T()>::type  
    dispatchSync( T fn );
```

Änderungen für MSVS 2022 und C++2x (latest) 2/5

- Datei: include\cinder\app\AppBase.h
- Zeile 615/616

```
template<typename T>  
typename std::result_of<T()>::type  
    AppBase::dispatchSync( T fn )
```

=>

```
template<typename T>  
typename std::invoke_result<T()>::type  
    AppBase::dispatchSync( T fn )
```

Änderungen für MSVS 2022 und C++2x (latest) 3/5

- Datei: include\cinder\app\AppBase.h
- Zeile 621

```
typedef typename std::result_of<T()>::type result_type;
```

=>

```
typedef typename std::invoke_result<T()>::type result_type;
```

Änderungen für MSVS 2022 und C++2x (latest) 4/5

- Datei: include\cinder\Filesystem.h
- Zeile 40

```
namespace fs = std::experimental::filesystem;
```

=>

```
namespace fs = std::filesystem;
```

Änderungen für MSVS 2022 und C++2x (latest) 5/5

- Verzeichnis-Namen ändern

`lib\msw\x64\Debug\v140` => `lib\msw\x64\Debug\v143`

`lib\msw\x64\Release\v140` => `lib\msw\x64\Release\v143`

`lib\msw\x86\Debug\v140` => `lib\msw\x64\Debug\v143`

`lib\msw\x86\Release\v140` => `lib\msw\x64\Release\v143`

Cinder

▪ **Lib für:**

- Graphics (OpenGL)
- Audio
- Video
- Networking
- Image processing
- Computational Geometry

▪ **Enthält u.a.**

- Boost – immer eingebunden
- Box2D
- Cairo
- FMOD
- OpenCV
- Open-Sound-System (OSC)

- **Natürlich könnte man auch andere Libs verwenden**
 - Cairo – Teil von Cinder – <http://cairographics.org/>
 - Open-Framework – <http://www.openframeworks.cc/>
 - Qt – <http://qt-project.org/>
 - SFML – <http://www.sfml-dev.org/>
 - Unreal Engine 4 – <https://unreal-engine-4.zeeff.com/tom.looman>
 - Und sicher gibt es auch noch weitere...
- **Ich bin einfach mal durch Zufall auf Cinder gestoßen worden**
 - C++Now Video von Herb Sutter
 - Und mir gefiel Cinder gut und kam gut damit zurecht
 - Und meine Plattformen (Windows, Mac OS X, iOS) sind abgedeckt
- **Ich kenne die anderen Bibliotheken nicht**
 - Vielleicht sind sie besser, keine Ahnung

Achtung - Disclaimer

- **Dies ist keine Cinder-Schulung**
 - Wir werden Cinder einfach nur ganz pragmatisch nutzen
 - Ohne ins Detail von Cinder und OpenGL und so einzugehen
- **Ich bin auch kein Cinder Experte**
 - Ich habe Cinder einfach nur ein bisschen in “Spiel-Projekten” genutzt
 - Habe mir ein Tutorial durchgelesen
 - Habe mir ein paar der beiliegenden Beispiele näher angeschaut
 - Mehr nicht

Wir nutzen hier heute in der C++ Kochshow

- **Microsoft Visual Studio 2022 (für C++2x latest)**
- **Cinder Version 0.9.2 mit Anpassungen**
- **Mehr nicht**

Achtung – noch ein Disclaimer

- **Wir werden “echten” C++ Quelltext sehen**
 - An manchen Stellen ist es Stil-Frage wie man Dinge löst
 - Manche Dinge sind historisch entstanden
 - Ich habe leider nicht mehr alle Dinge konsistent und “optimal” ändern können
 - Es sind auch noch ein paar Fehler bzw. Unschönheiten im Programm
 - Mal sehen, ob Sie sie finden
 - Man kann sicher manches besser lösen
 - Und natürlich abgedrehter und allgemeiner lösen
 - Wer solche Stellen findet, darf sie behalten
 - Trotzdem dürfen wir natürlich darüber diskutieren
 - Und ich lerne ja auch gerne dazu
 - Aber das ist nicht das primäre Thema heute

Wie setzt man am einfachsten ein Cinder-Projekt auf?

▪ **TinderBox nutzen**

- tools\TinderBox-Win\TinderBox.exe
- TinderBox erstellt das Projekt mit allen Abhängigkeiten
 - Projekt erstellen für Microsoft Visual Studio 2015
 - Beim Laden ins Visual Studio das Projekt updaten lassen
 - Umstellen des Projekts auf C++2x
- Weitere Libs können eingebunden werden
 - Siehe Live-Demo
- Git-Repository kann direkt mit angelegt werden

▪ **Erstellt**

- Einfaches Programm mit schwarzem Hintergrund-Fenster

▪ => **TinderBox Demo**

Was lernen wir vom Initial-Code?

- **Klasse „CinderBeispielApp“**
 - Objekt, das die Haupt-Anwendung repräsentiert
- **Funktion „setup()“**
 - Wird einmal am Anfang aufgerufen, damit die Anwendung sich initialisiert
 - Die Cinder-Infrastruktur steht hier schon zur Verfügung
 - Im Gegensatz z.B. zur Funktion „prepareSettings(Settings* settings)“, die noch früher aufgerufen wird
- **Funktion „update()“**
 - Diese Funktion wird n-mal die Sekunde aufgerufen (Frame-Rate) aufgerufen
 - Hier soll sich die Anwendung auf den neuen Zustand updaten
 - **Nicht zeichnen**
 - Hinweis: die Frame-Rate kann in den Settings eingestellt werden – siehe Funktion "prepareSettings(Settings* settings)“

Was lernen wir vom Initial-Code?

▪ Funktion „draw()“

- Auch diese Funktion wird n-mal die Sekunde aufgerufen (Frame-Rate) aufgerufen
- Immer nach „update“
- Könnte ausgelassen werden, wenn die Frame-Rate nicht erreicht wird
- Hier soll sich die Anwendung neu ausgeben

▪ CINDER_APP

- Dieses Makro enthält das entsprechende „main“ der Zielplattform und erzeugt das Anwendungs-Objekt

▪ **mouseDown(MouseEvent event)**

- Beispiel für das Event-Handling in Cinder
- Konkret wird diese Funktion aufgerufen, wenn ein Maus-Button gedrückt wird
- Dazu später mehr

Cinder-App erwartet optional ein Callable

■ Zum Beispiel

- Lambda (siehe Cinder02 Beispiel)
- Klassenfunktion (siehe weitere Cinder Beispiele)

■ Signatur

- (App::Settings*)

■ Was kann man da z.B. machen?

- `settings->setTitle("Cinder-Applikation");`
- `settings->setFullScreen();`
- `settings->setResizable(false);`
- `settings->setWindowPos(100, 100);`
- `settings->setWindowSize(1024, 768);`
- `settings->setFrameRate(30.0f);`

■ => Cinder02

Raumschiff "malen"

- Fangen wir an, mal das Raumschiff zu "malen"
- Eigentlich sollte man ein schönes Bild nehmen, aber heute im Kochstudio sind wir ganz pragmatisch
- Wir nehmen einfach mehrere Kreise, die überlappend gezeichnet werden
- Kreise zeichnen – fertiger Befehl in Cinder
- => Cinder03

Raumschiff "malen"

- **Überlappende Kreise**
- **Auslagern in**
 - Eine Klasse "Ship"
 - Klassen-Funktion "Draw"
- **=> Cinder04**

Start-Raumschiff ins Spiel integrieren

- **Klasse Ship zum Leben bringen**
 - Schiff mit Ort und Richtung
 - Farbe
 - Bildschirm-Größe
 - Smart-Ptr
- **Integrieren ins Spiel**
 - Erzeugen in der Bildschirm-Mitte mit Richtung nach oben
 - Zeichnen nur, wenn Schiff vorhanden ist
- => **Cinder05**

Raumschiff rotieren

- **Rotate Funktion ins Schiff integrieren**
- **Abfrage der seitlichen Cursor-Tasten**
 - Nicht direkt reagieren, sondern nur Bool-Flags setzen
 - Dann in Update bearbeiten
 - Soll man so in Cinder machen
 - Könnte sonst Probleme machen, wenn es gleichzeitig mit “update” oder “draw”
 - In “update” auswerten
 - Achtung – aufpassen, wenn Schiff nicht vorhanden
- => **Cinder06**

Raumschiff beschleunigen

- **Abfrage der Up/Down Cursor-Tasten**
 - Behandlung wie bei der Rotation
- **Beschleunigungs-Funktion ins Schiff integrieren**
- **Speed dann in Ship.update auswerten**
 - Ort in Richtung Direction mit Speed verschieben
 - Achtung
 - Float Rechnungen landen nicht zwingend wieder bei "0"
 - Darum wird Speed unter einer Schwelle auf "0" gesetzt
 - Eine Art "Reibung" einbauen
- => **Cinder07**

Spielfläche soll "unendlich" sein

- **Ränder gehen auf der anderen Seite wieder rein**
- **Ship.update muss das berücksichtigen**
 - Ort normieren, wenn außerhalb des Spielfelds
- **=> Cinder08**

Kantenbehandlung

- **Zeichnen direkt am Rand muss auch auf der anderen Seite erfolgen**
 - Schiff mit Funktionen für Radius und Größe erweitern
 - In Ship.draw dann – wenn notwendig – mehrfach zeichnen
 - Achtung – ich habe es schon selber bemerkt
 - Die Lösung ist nicht vollständig
 - Was muss noch zusätzlich gemacht werden?

- => **Cinder09**

Was ist ein Raumschiff ohne Waffen – Feuer!

▪ **Torpedo**

- Wird einfach durch einen kleinen roten Kreis ausgedrückt
- Klasse “Torpedo” quasi analog zu “Ship”
 - Copy & Paste Programmierung
 - Refactoring kommt später
 - Erstmal Schritt für Schritt zum Laufen bringen

▪ **Das Schiff kann jetzt feuern**

- Neue Funktion, die ein erzeugtes Torpedo-Objekt zurückgibt
 - Mit korrekter Location und Direction (übernommen vom Schiff)

Was ist ein Raumschiff ohne Waffen – Feuer!

- **App**
 - Tastatur-Steuerung zum Feuern (“x”)
 - Vector von Torpedo-Objekten
 - Alle Torpedo-Objekte updaten und drawen

- => **Cinder10**

Die Reichweite ist unendlich

- **Der Schuß endet nie, das darf so nicht sein**
 - Damit man sich nicht in den Rücken schießt
 - Nur 0,9 x Bildschirm-Höhe
- **Zurückgelegte Strecke in Torpedo integrieren**
 - mLength
 - In Torpedo.update berücksichtigen
- **Was, wenn die Strecke vorbei ist?**
 - Das Torpedo-Objekt kann sich ja nicht selber löschen
 - Update gibt zurück, ob das Objekt weiter lebt oder nicht
 - App muss das dann in “update” berücksichtigen
 - For_each => remove_if und erase
- => **Cinder11**

Refactoring von Torpedo und Ship

■ Gemeinsamkeiten in Basis-Klasse “Figure” verschieben

- Screen-Größe
- Location & Direction
- Update mit Verlassen-Feld Behandlung
- Draw mit Kanten-Mehrfach-Zeichnen Behandlung
- Abstrakte Funktionen für
 - size
 - update und draw
 - Natürlich mit Idiom: virtual nie public, public nie virtual
 - => doUpdate und doDraw

■ Weitere Änderungen

- Ship.update gibt jetzt auch “bool” zurück – immer true
- App setzt nur in Figure Screen-Größe
- Abgeleitete Klassen haben Super-Typedef Idiom

■ => Cinder12

Schiffs-Explosion

- **Neue Klasse “ExplosionShip”**
 - Ähnlich Torpedo und Ship
 - Lebt 50 Frames
 - Explosion ist auch nur ein Kreis, aber ein dynamischer
 - Bläht sich 25 Frames auf, dann schrumpft er 25 Frames
 - Von Radius 0 zu Radius 50 und wieder zurück zu Radius 0
 - Und er verändert dabei seine Farbe (Rot-Ton)
 - Danach gibt “update” false zurück, und die Explosion ist Geschichte
- **App baut einen “ExplosionShipPtr” ein**
 - Erstmal ohne Wechselwirkung
 - Wird ausgeführt statt “Feuer” – Taste “x”
- **=> Cinder13**

Asteroiden Explosion

- **Wenn wir schon mal mit Sprengstoff spielen**
 - Dann können wir auch direkt die Asteroiden Explosion erstellen
- **Analog zu ExplosionShip**
 - Copy & Paste
 - Nur kleiner, und schneller vorbei
 - Max-Radius 10
 - 20 Frames Lebensdauer
 - Andere Rot-Töne
- **Einbau in App zum Testen wie eben**

- => Cinder14

Refactoring der Explosionen

- **Gemeinsamkeiten in Basis-Klasse “Explosion” ziehen**
- **Konkrete Klassen enthalten nur noch**
 - doUpdate
 - Wachstums-Prozeß und Lebensdauer
 - doDraw
 - Farbveränderung und Zeichnen
 - Max Größe als Konstante
- **App**
 - Von einer Explosion
 - Member ExplosionPtr
 - Zu vielen
 - Vector<ExplosionPtr>
 - Test-Code beim Feuern bleibt, nur nun für 2 Explosionen
- **=> Cinder15**

Kollision – „Friendly Fire“

- **Das Schiff schießt sich selbst ab**
 - Das geht immer noch, trotz Schuß-Reichweiten-Begrenzung
- **Figure**
 - Neu: Abfrage des aktuellen Orts
 - getLocation
- **App**
 - Teil des Update-Prozesses
 - Nach dem sich alle bewegt haben – Kollisions-Check
 - Hier erstmal nur Friendly-Fire
 - Dabei können Figuren entfallen (zerstört) – Torpedo, Schiff, Asteroiden
 - Und andere hinzukommen (neu) – Asteroiden, Explosionen
- => **Cinder16**

Bugfix und etwas Refactoring

- **Es ist noch ein Bug vorhanden**
 - Sehen Sie ihn?
 - ???????????
 - ???????????
- **Refactoring**
 - Immer die gleichen Arbeiten auf den Containern
 - Z.B. `remove_if` und `erase`, `for_each` und `copy`
 - In eigene Funktionen auslagern
 - Neuer Header `Utils`
 - Macht App einfacher
 - Würde man modern mit `Ranges` in C++20 lösen
 - Hatte noch keine Zeit für einen Umbau - sorry

Bugfix und etwas Refactoring

- **Es ist noch ein Bug vorhanden**
 - Schiff schießt sich selbst ab beim Vorwärts-Bewegen
 - Feuer materialisiert im Schiff selber
 - => Ein update Vorgang im Torpedo-Konstruktor
- **Refactoring**
 - Immer die gleichen Arbeiten auf den Containern
 - Z.B. `remove_if` und `erase`, `for_each` und `copy`
 - In eigene Funktionen auslagern
 - Neuer Header `Utils`
 - Macht App einfacher
 - Würde man modern mit `Ranges` in C++20 lösen
 - Hatte noch keine Zeit für einen Umbau - sorry
- => **Cinder17**

Asteroiden einführen

- **Im Prinzip wie Torpedo, Ship & Explosion**
 - Grauer Kreis
 - 4 Größen: Huge, Big, Medium & Small
 - Kann von außen gewählt werden – Konstruktor
 - Unterschiedliche Größen
 - 4 Entstehungs-Regionen: LT, RT, LB, RB
 - Kann von außen gewählt werden – Konstruktor
 - Genauer Ort wird per Zufall gesetzt
 - Direction & Speed
 - Wird per Zufall gesetzt
 - Richtung immer grob Richtung Spiel-Mitte
- **App**
 - Bekommt `Vector<Asteroiden>`
 - Beim Start wird je Region ein Asteroid unterschiedlichen Typs gesetzt
- **=> Cinder18**

Asteroiden Kollisionen mit dem Schiff

- **Asteroid braucht noch Radius Funktion für Kollisions-Berechnung**
- **App**
 - Erzeugen nun nur noch Hugel-Asteroiden
 - Kollisions-Behandlung
 - Kollisions-Erkennung
 - Schiff ist weg
 - Asteroid explodiert – ohne Reste
 - ExplosionShip bleibt übrig
- **=> Cinder19**

Treffer

- **Asteroid zerfällt in kleinere Asteroiden**
 - Small Asteroid ist weg
 - Funktion “hit” in Asteroid
 - Nette Anwendung von switch ohne break
 - Asteroiden müssen an konkreter Location mit beliebigem Speed und Direction erzeugt werden => weiterer Konstruktor
- **App**
 - Kollisions-Behandlung
 - Treffer-Erkennung
 - Torpedo ist verbraucht
 - Asteroid explodiert – mit Resten (neue Asteroiden)
 - ExplosionAsteroid bleibt auch übrig
- => **Cinder20**

Text-Darstellung in Cinder / OpenGL

▪ Text-Ausgabe

- Font erzeugen
- => TextureFont erzeugen
- Farbe “global” setzen
- ÜberTexture-Font Text an Location ausgeben

▪ Außerdem

- `glDisable(GL_TEXTURE_2D)`
- `enableAlphaBlending()`
- Keine Ahnung, warum und weshalb, muss aber so sein

▪ => Cinder 21

Punkte zählen

▪ Punkte

- Asteroid bekommt Punkte je nach Typ (Größe)
- Asteroid gibt Punkte zurück, die er “wert” ist
 - => 10, 30, 60, 100

▪ App

- Member für
 - Punkte
 - Text-Darstellung
- Setup setzt Font und TextureFont
- Kollisions-Behandlung summiert Punkte auf
- Update zeigt Punkte links oben an

▪ => Cinder22

Mehrere Leben (Schiffe)

▪ App

- Zähler für Schiffe
- Taste für neues Schiff ("n")
- Update behandelt Neues-Schiff-Taste
- Draw zeichnet die noch vorhandenen Schiffe rechts oben hin
 - Jetzt rentiert sich die Klassen-Funktion Draw

▪ => Cinder23

"Cheftaste"

- Taste "q" für Programm-Ende
- Cinder hat "quit()" dafür

- => Cinder24

Fertig

Viel Spaß beim Spielen

Fragen?