PROGRAMMIERSPRACHEN

Dozent: Dr. Andreas Jäger

Programmiersprachen - Formale Sprachen

Beschreibung von:

Algorithmen, Datenstrukturen

so dass diese von einem Computer ausgeführt werden können.

Erlaubt die Verwendung/- besitzen unterschiedlicher Abstraktionslevel.

Wie natürliche Sprachen besitzen sie eine Syntax(Form) und Semantik(Bedeutung).

Programmiersprachen

Wir unterscheiden *low-level Sprachen* und *high-level Sprachen* und *visuelle Sprachen*.

low-level Sprachen:

Früher wurden Programmiersprachen noch für eine bestimmte Rechnerarchitektur gebaut.

Das level der Abstraktion ist relativ gering.

Bsp.: Assembler

Programmiersprachen

Wir unterscheiden *low-level Sprachen* und *high-level Sprachen* und *visuelle Sprachen*.

Low-level Sprachen:

Assembler

High-level Sprachen:

BASIC, C(++), C#

Visuelle-Sprachen:

MVPL, UE4, Unity

Programmiersprachen

Low-level Sprachen:

Bsp.: Assembler

Ist nicht „eine“ Sprache – jede low-level Sprache, die eine starke Verbindung zwischen dem Architekturspezifischen Maschinencode und der Programmsprache besitzt.

Das level der Abstraktion ist relativ gering.

Programmiersprachen

Zwischenfrage?

Welche beiden Chiparchitekturen sind heutzutage dominant?

CISC (Complex instruction set computer) ARM(Advanced RISC Machine)

Programmiersprachen

High-level Sprachen:

Bsp.: C(++)

Hohes Abstraktionslevel von der Architektur des Computers.

Kann eine Syntax und Semantik verwenden, die Elemente von natürlichen Sprachen verwendet.

Python:

```
for elment in vector:
    extract(element)
```

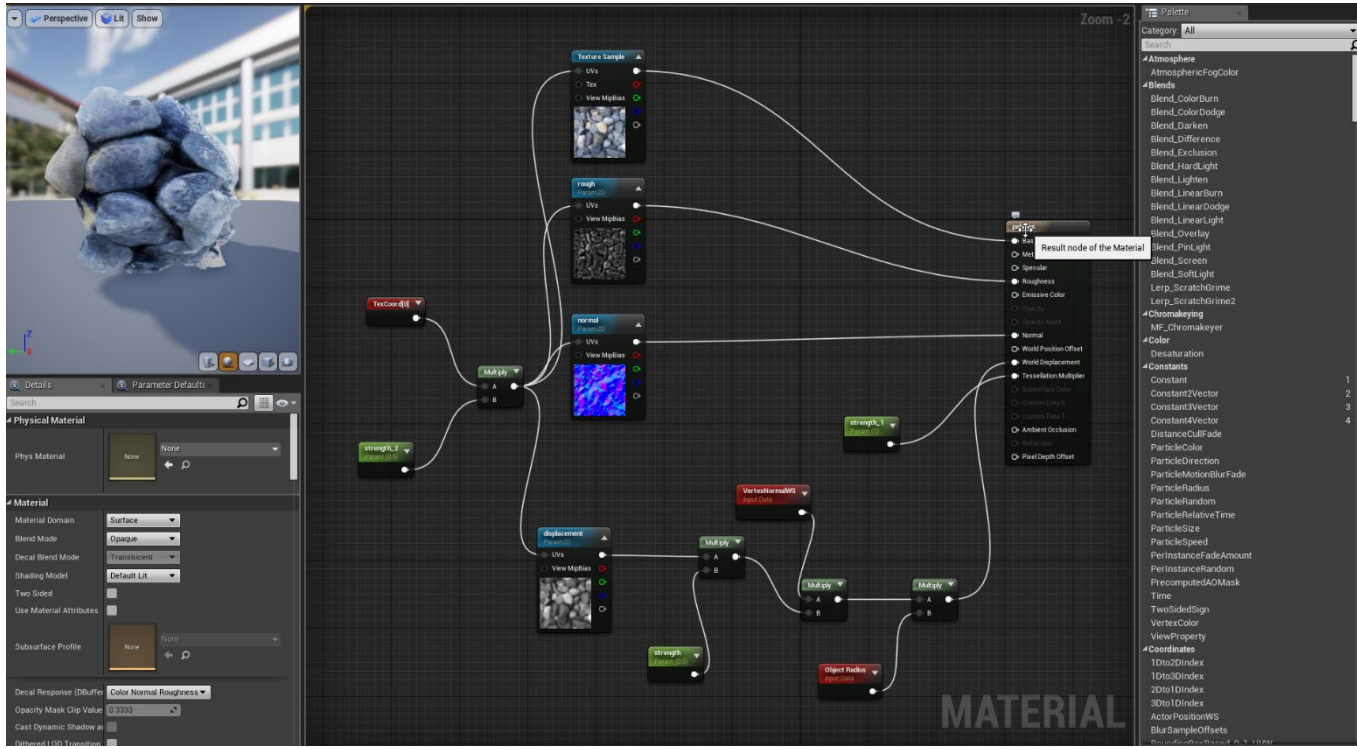
Programmiersprachen

Visuelle-Sprachen:

Bsp.: UE4

Programme können durch die graphische Manipulation von Programmierungselementen erstellt werden.

Alle Visuelle Programmiersprachen benötigen heutzutage noch zusätzlich andere Sprachen.

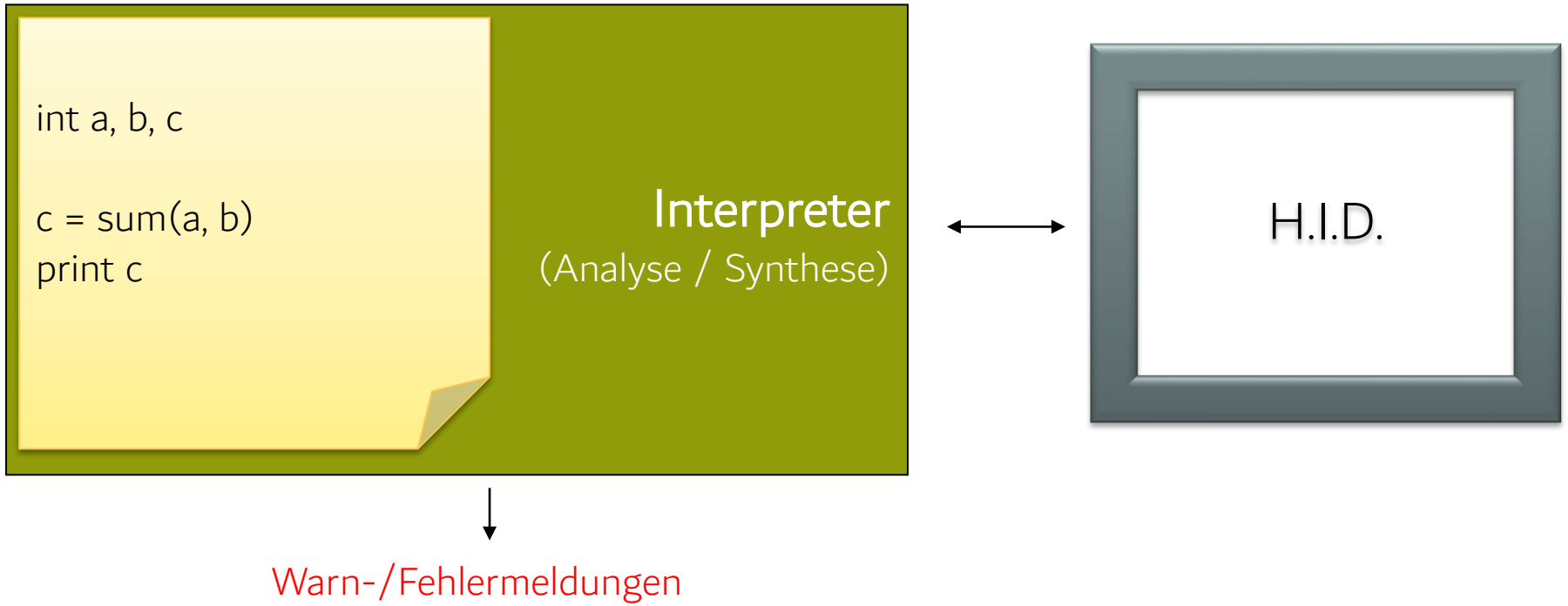


INTERPRETER VS COMPILER

Dozent: Dr. Andreas Jäger

Interpreter

Beim Ausführen eines Programmes werden die einzelnen Zeilen des Programmes beim Programmstart gelesen, analysiert und dann ausgeführt. Code wird zur Laufzeit gelesen und interpretiert.



Interpreter

Vorteil:

Direkte Programmausführung.

Manipulation des Codes zur Laufzeit.

Nachteil:

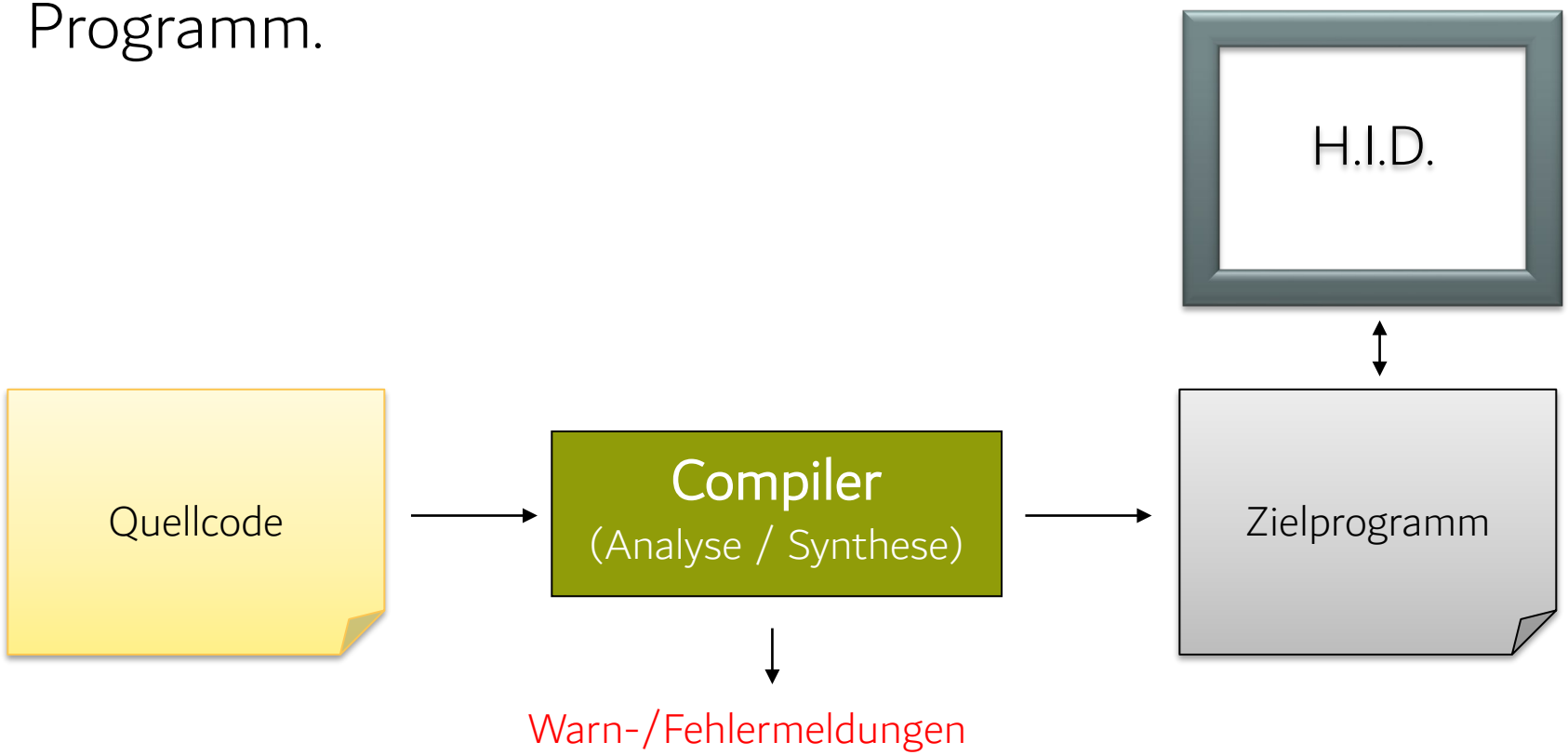
Laufzeiten sind oft höher.

Laufzeit Optimierungen, die ein Compiler durchführen kann sind kaum möglich.

Fehler warden erst zur Laufzeit bemerkt.

Compiler

Der Programmcode kann nicht direkt ausgeführt werden. Der Compiler analysiert den kompletten Code und erzeugt daraus ein ausführbares Programm.



Compiler

Vorteil:

Performanter, weil schon im Voraus optimiert werden kann.
Viele Programmierfehler werden durch Compiler schon vorab erkannt.

Nachteil:

Nur umständlich zur Laufzeit dynamisch manipulierbar

Warum C/C++

Leistungsfähige Hochsprache.

Offener Standard (ANSI).

Viele Sprachen haben sich an der C Syntax orientiert.

C#, Java, PHP…

Wer C kann, kann viele Sprachen leichter lernen.

Unterstützt mehrere Programmierparadigmen.

VIELEN DANK
FÜR
IHRE AUFMERKSAMKEIT!