

Architektur von Videospielen

Nico Hauff, Vincent Langenfeld, Frank Schüssele

November 10, 2022

Beginnen Sie jetzt mit dem Programmieren

- ▶ z.B.:
`/<spielname>/<spielname>.sln`
- ▶ [Jenkins](#) und [Sonar](#) sind nun online
- ▶ Arbeiten Sie an [einem](#) Projekt, integrieren Sie [kontinuierlich](#)
- ▶ [MS01](#): Nächste Woche (Spielobjekt in der Welt bewegbar, bewegliche Ansichten, Level laden/speichern, Soundausgabe)
- ▶ Nehmen Sie die wöchentlichen Aufgaben als Chance sich zu verbessern

Ideenpräsentation (Do. Nächste Woche)

- ▶ Ab 14 Uhr ct. [hier](#)
 - ▶ Ab 14 Uhr st. Technik testen
 - ▶ Gesamte Gruppe
- ▶ max 10min (+ 5min Fragen)
- ▶ Bereiten Sie sich vor:
 - ▶ Worum geht es?
 - ▶ Zentrale Spielmechanik?
 - ▶ Spielablauf? (gewinnen, verlieren)
 - ▶ Warum macht es Spaß?

Architektur

Engine

- ▶ Eingabe
- ▶ Menüs und Popups
- ▶ Persistente Einstellungen
- ▶ Speichern/Laden
- ▶ Rendern und Kamera
- ▶ Pathfinding
- ▶ Kollisionserkennung
- ▶ Netzwerk
- ▶ Objektverwaltung

Spielmechanik

- ▶ Spielobjekte und Eigenschaften
- ▶ Interaktion
- ▶ KI

Tools

- ▶ Modelle integrieren
- ▶ Texturen integrieren
- ▶ Sounds integrieren
- ▶ Level erstellen
- ▶ Debugging output

Content

- ▶ Sound und Musik
- ▶ Modelle
- ▶ Sprites
- ▶ Levels

Engine

- ▶ Eingabe
- ▶ Menüs und Popups
- ▶ Persistente Einstellungen
- ▶ Speichern/Laden
- ▶ Rendern und Kamera
- ▶ Pathfinding
- ▶ Kollisionserkennung
- ▶ Netzwerk
- ▶ Objektverwaltung

Spielmechanik

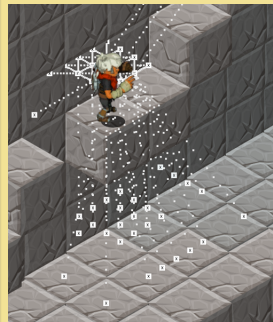
- ▶ Spielobjekte und Eigenschaften
- ▶ Interaktion
- ▶ KI

Tools

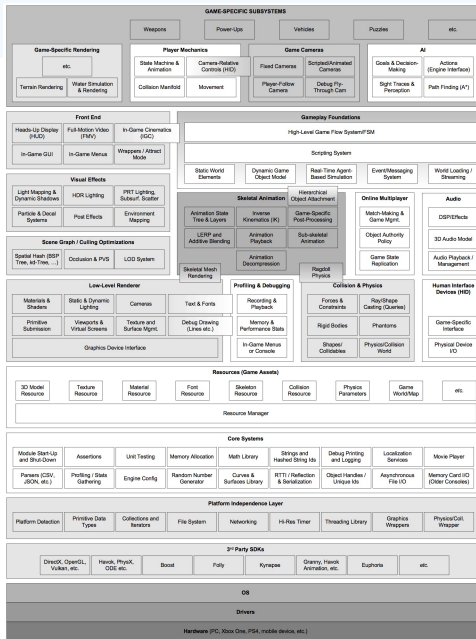
- ▶ Modelle integrieren
- ▶ Texturen integrieren
- ▶ Sounds integrieren
- ▶ Level erstellen
- ▶ Debugging output

Content

- ▶ Sound und Musik
- ▶ Modelle
- ▶ Sprites
- ▶ Levels



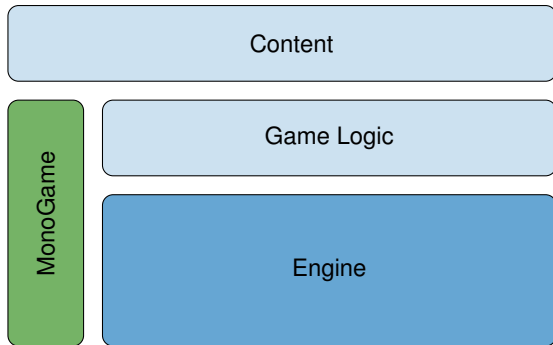
A Man Ran (WS17/18)



Gregory, Game Engine Architecture, 2019

Am Besten **Top-Down** anfangen:

- ▶ Zuerst Struktur aufbauen
- ▶ Funktionalität aufteilen
- ▶ Separation of Concerns



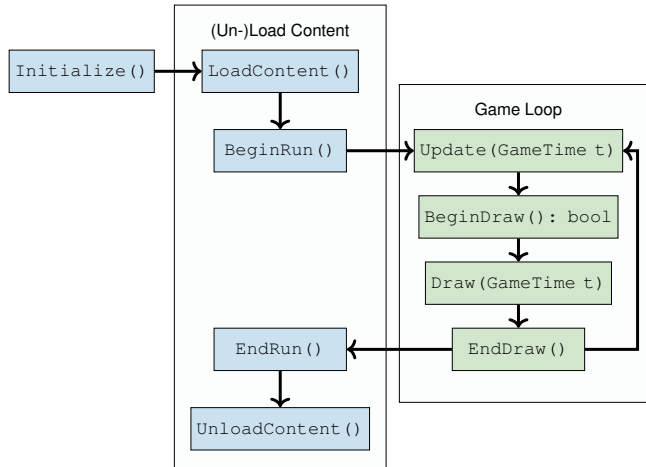
MonoGame und .NET

.net

- ▶ Mathematik
- ▶ Zufallszahlen
- ▶ (De-)Serialisierung in Json/XML/Binär
- ▶ Datenstrukturen (Liste, Menge, ...)
- ▶ Debugging
- ▶ Profiling

MonoGame

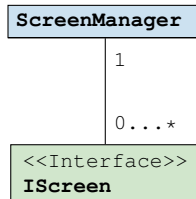
- ▶ Abstraktion
 - ▶ Grafik
 - ▶ Sound
 - ▶ Input
- ▶ Content Pipeline
- ▶ Einfache Anzeigemethoden (BasicEffect, SpriteBatch)
- ▶ Datentypen
 - ▶ Vector2, Vector3
 - ▶ Matrix
 - ▶ ...
- ▶ Game Loop

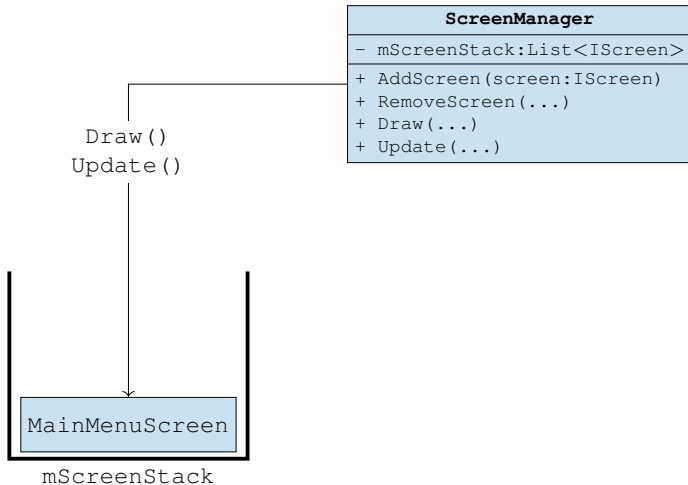


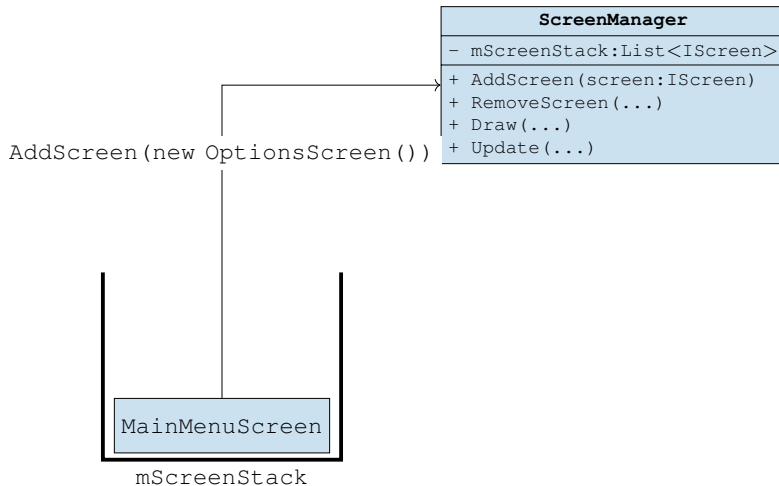
Bausteine

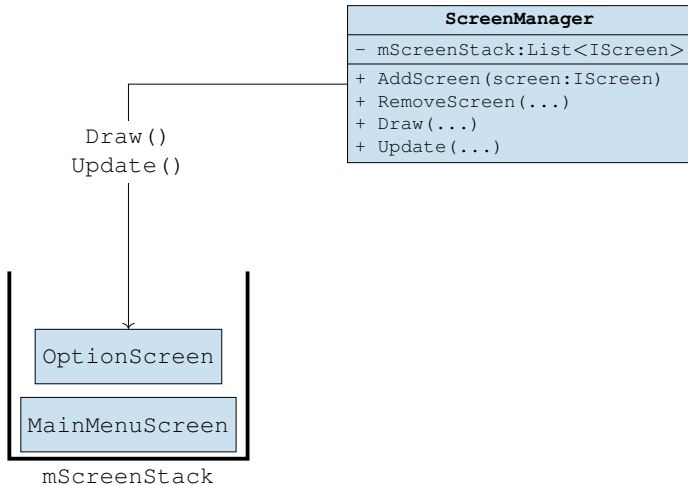
Ziel: Entkoppeln von Verwaltung und Implementierung von Spielzuständen

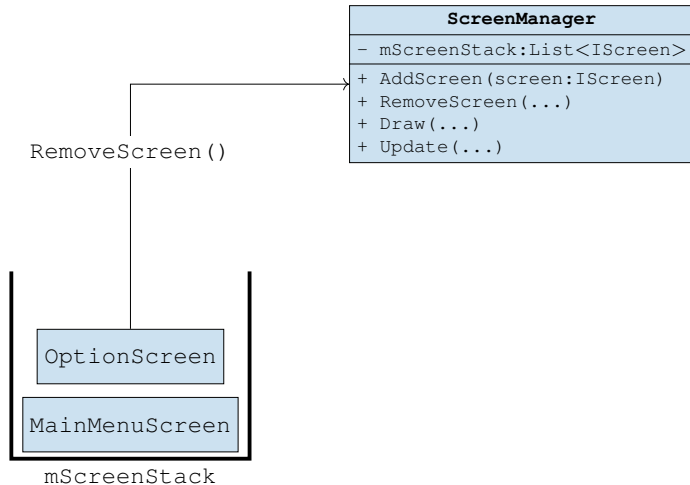
- ▶ **Screens** implementieren ihre jeweilige Funktionalität
 - ▶ Hauptmenü
 - ▶ Optionsmenü
 - ▶ Spielansicht
 - ▶ HUD
 - ▶ Ladebildschirm
 - ▶ ...
- ▶ **ScreenManager** verwaltet alle Screens
 - ▶ Kontrollfluss
 - ▶ Hinzufügen
 - ▶ Entfernen

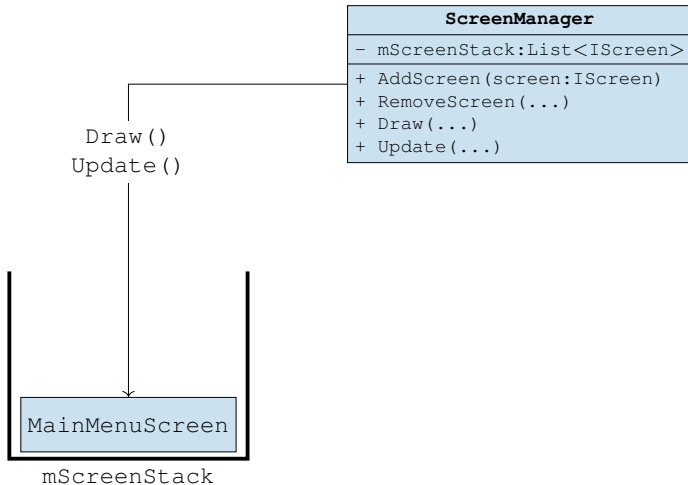


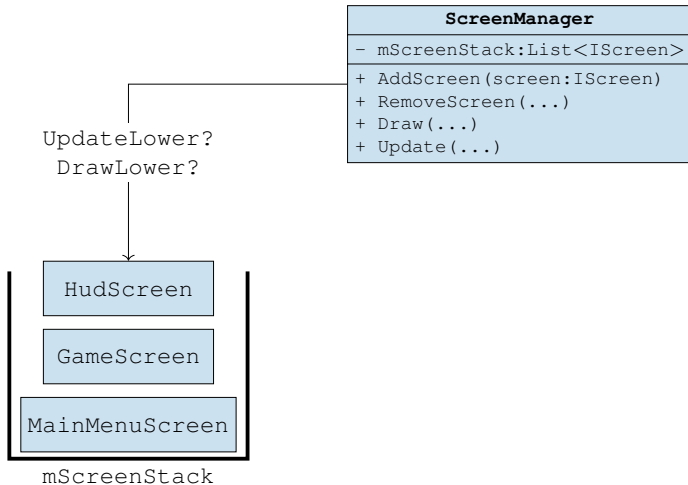


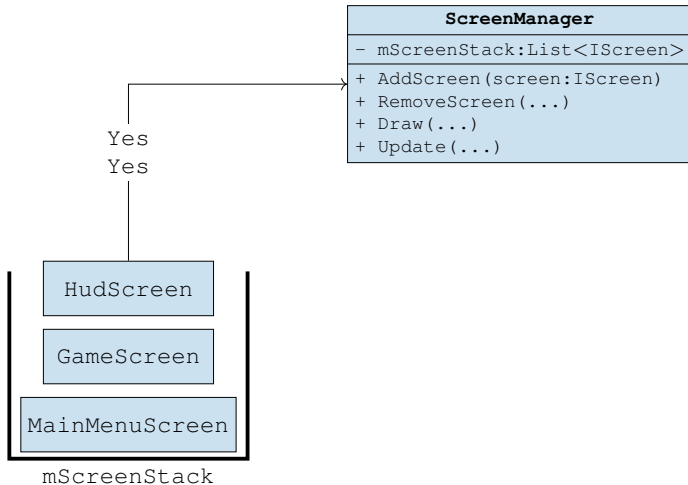


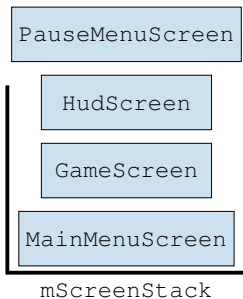




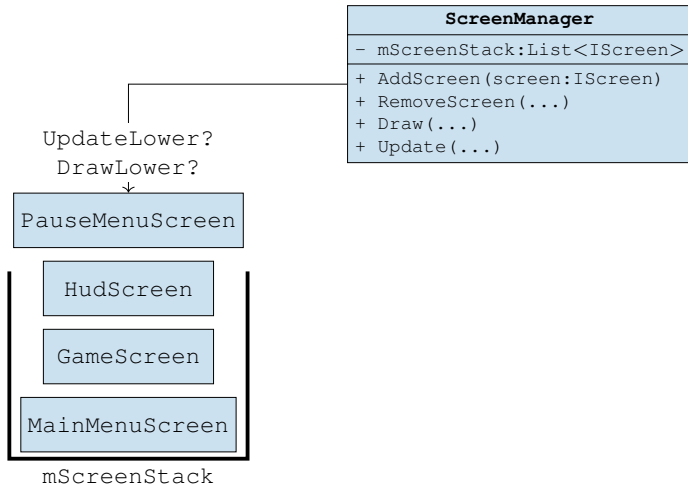


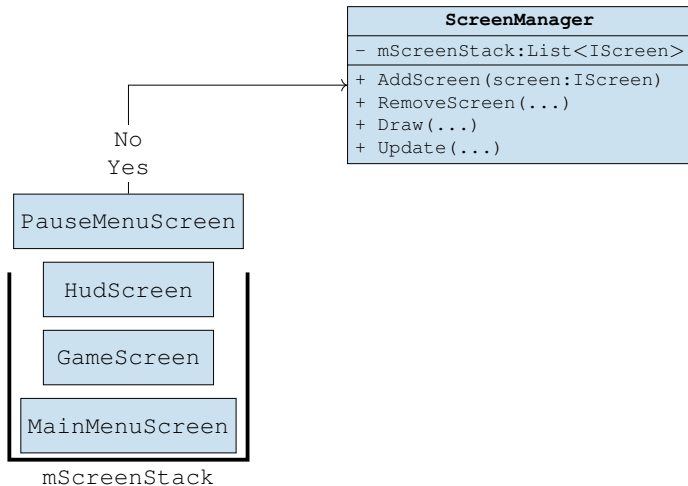


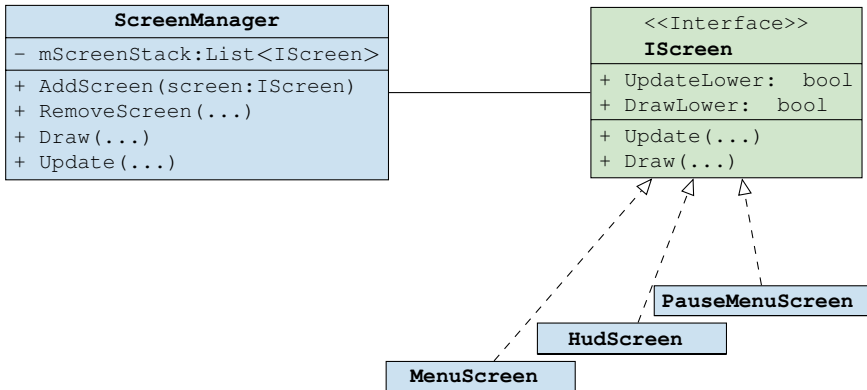




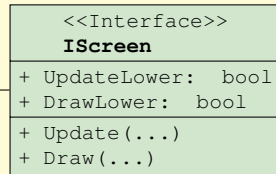
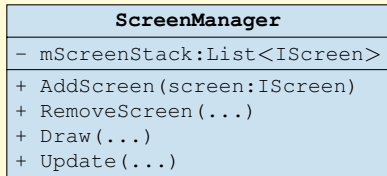
ScreenManager
- mScreenStack:List<IScreen>
+ AddScreen(screen:IScreen)
+ RemoveScreen(...)
+ Draw(...)
+ Update(...)



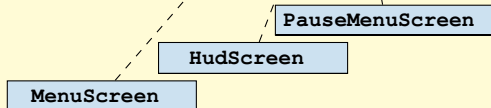




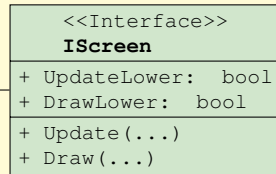
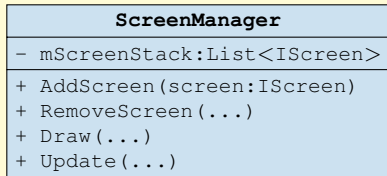
Engine.ScreenManagement



GameLogic.Menu



Engine.ScreenManagement



<<component>>
ScreenManagement

IScreen

<<component>>
Menu

Menu

PauseMenuScreen

HudScreen

MenuScreen

- ▶ Tastatur, Maus, Gamepad, ...
- ▶ MonoGame Namespace `Microsoft.Xna.Framework.Input`
 - ▶ `KeyboardState mKeyboardState = Keyboard.GetState();`
 - ▶ Statusinformation per Frame: Zustand der Tasten, Mausposition
 - ▶ keine Historie
- ▶ Wir brauchen
 - ▶ Statusinformation in Abhängigkeit von Zeit
(z.B.: A wurde diese Frame losgelassen)
 - ▶ Abstraktion von Tasten zu Aktionen

Ziel: Abstraktion der konkreten physischen Eingaben (und Eingabegeräte)

`Xna...Mouse`

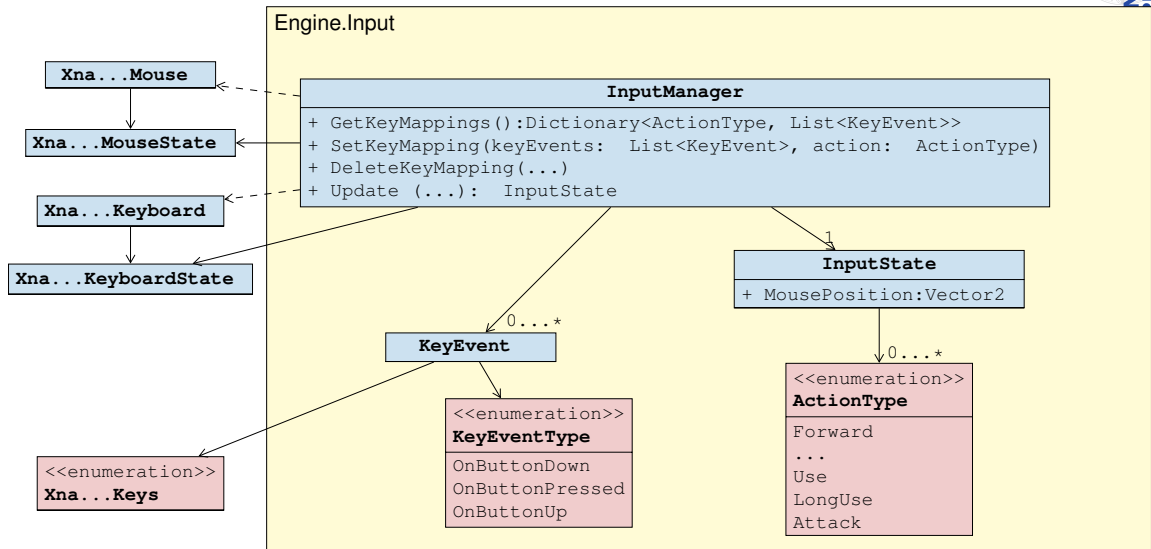
`Xna...MouseState`

`Xna...Keyboard`

`Xna...KeyboardState`

`<<enumeration>>`

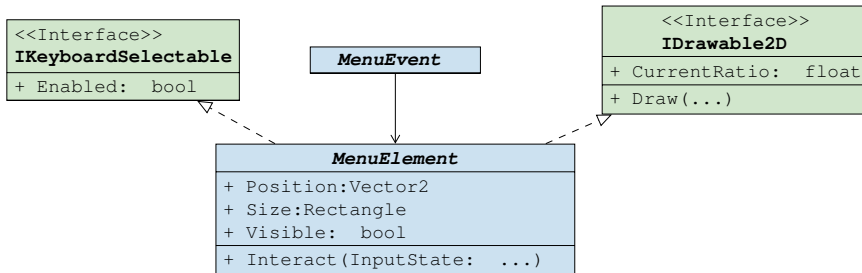
`Xna...Keys`

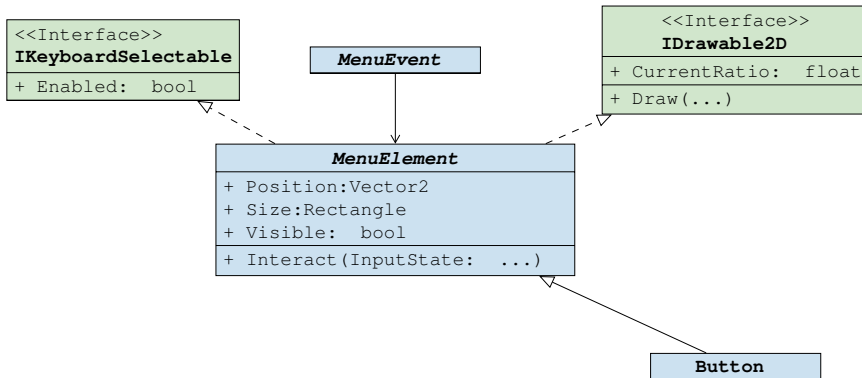


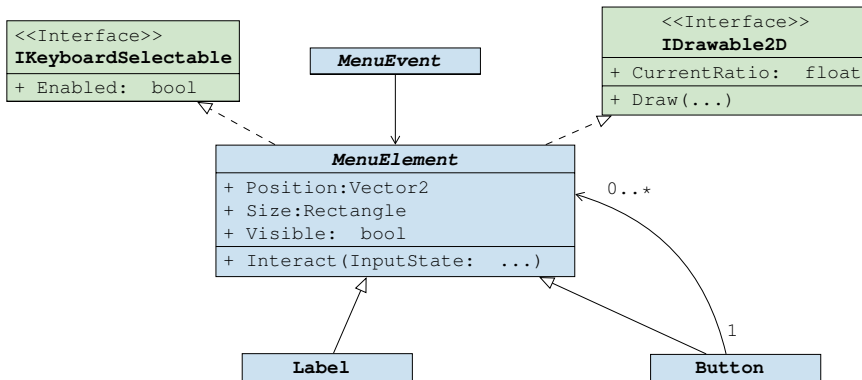
- ▶ Viele Elemente die sich sehr ähnlich verhalten
 - ▶ Panels
 - ▶ Buttons
 - ▶ Labels
 - ▶ Textboxen
 - ▶ Checkboxen
- ▶ Auch im Hud verwendbar

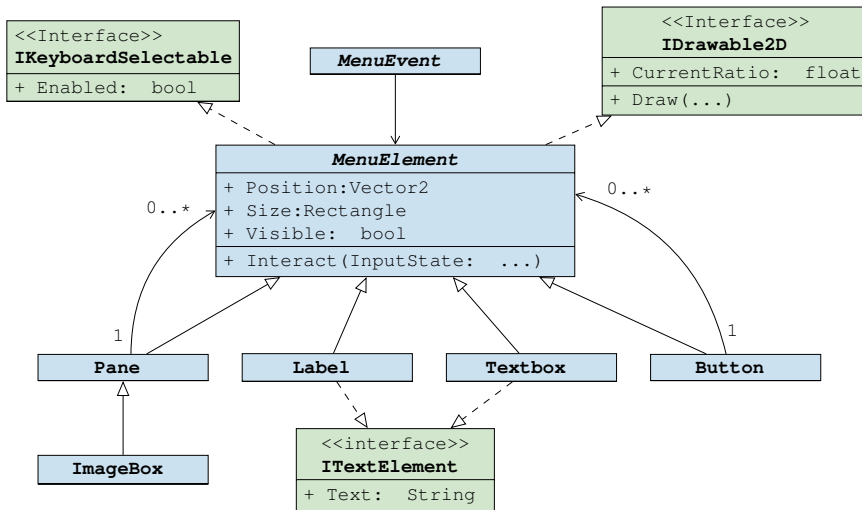
Ziel:

- ▶ Alle Eingabeelemente gleich verwalten
- ▶ Einfache Erweiterbarkeit

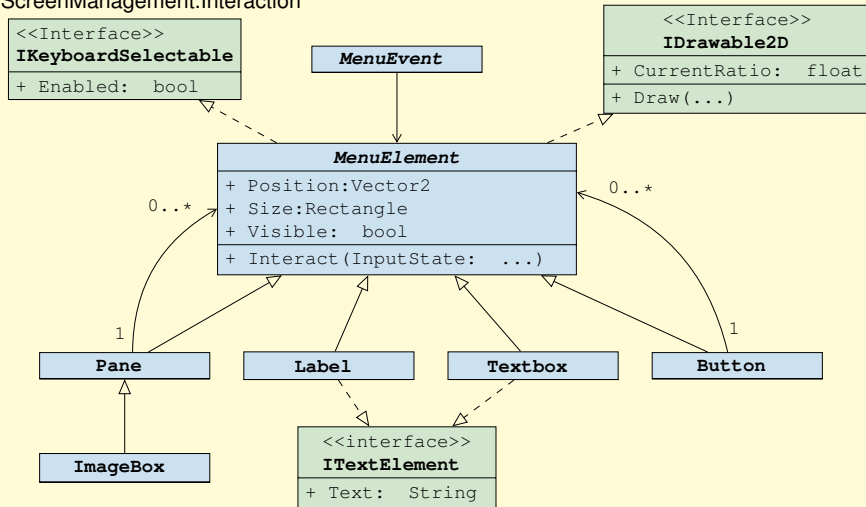








Engine.ScreenManagement.Interaction



Offline

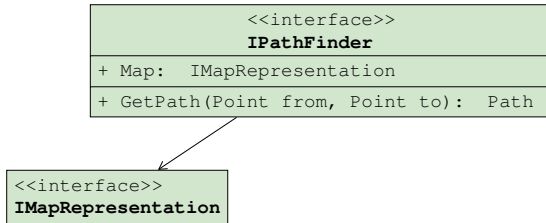
Auf dem kürzesten Weg von A nach B

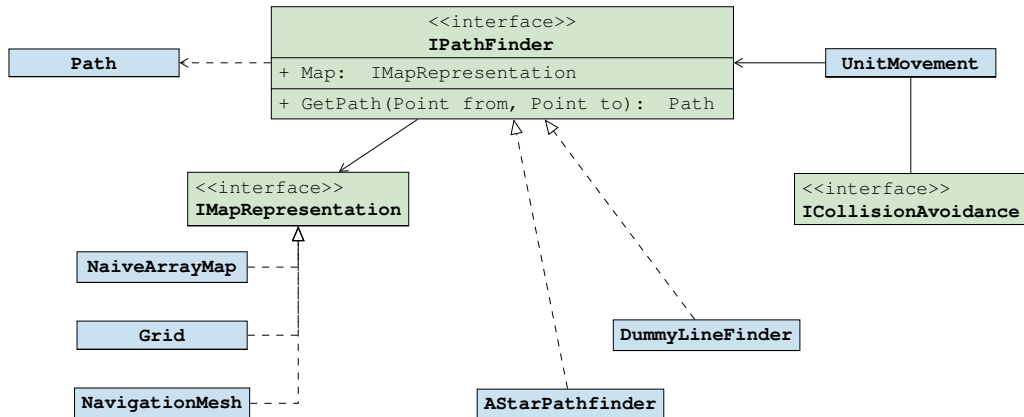
- ▶ Berücksichtigt Welt und unbewegliche Objekte
- ▶ Meistens A^*
- ▶ Viele mögliche Weltrepräsentationen:
Grid, Hierarchical Grid, Waypoint Graph,
Navigation Mesh,...

Online

Unterwegs mit niemandem zusammenstoßen

- ▶ Beweglichen Objekten ausweichen
- ▶ Verschiedene Verfahren:
Steering, Flocking, Flow Fields, ...





Spielobjekte

Spielobjekte sind alle Dinge, die eine Repräsentation in der Spielwelt haben.

- ▶ Charaktere, Fahrzeuge, Bäume, Raketen, Gras, Steine, Trigger, Lichter, Sounds, ...
- ▶ Spielobjekte müssen manchmal
 - ▶ gezeichnet werden
 - ▶ sich bewegen
 - ▶ zerstörbar sein
 - ▶ miteinander kollidieren
 - ▶ etc.
- ▶ Wie verwaltet man so viele (verschiedene) Objekte effizient?

Spielobjekte

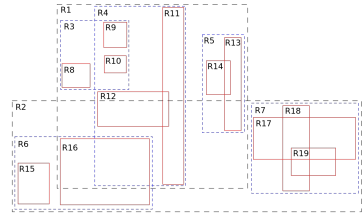
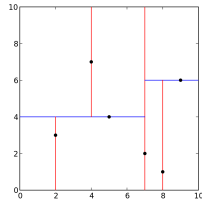
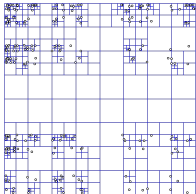
Spielobjekte sind alle Dinge, die eine Repräsentation in der Spielwelt haben.

- ▶ Charaktere, Fahrzeuge, Bäume, Raketen, Gras, Steine, Trigger, Lichter, Sounds, ...
- ▶ Spielobjekte müssen manchmal
 - ▶ gezeichnet werden
 - ▶ sich bewegen
 - ▶ zerstörbar sein
 - ▶ miteinander kollidieren
 - ▶ etc.
- ▶ Wie verwaltet man so viele (verschiedene) Objekte effizient?
 - ▶ Zur Laufzeit?
 - ▶ Im Softwaredesign?

Szenengraph

Ein Szenengraph ist eine zentrale Datenstruktur, die der logischen und räumlichen Verwaltung der Spielobjekte dient.

- ▶ Antwort auf räumliche Fragen
- ▶ Update- und Drawaufrufe an (relevante) Spielobjekte weitergeben
- ▶ Beispiele: Liste, Heap, Quad-/ Octree, KD-Tree, R-Tree, ...



Objektzentriert

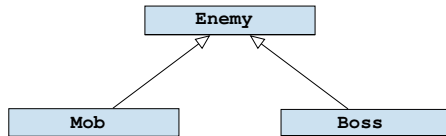
- ▶ Spielobjekte sind Klassen mit
 - ▶ Eigenschaften des Spielobjekts
 - ▶ Verhalten des Spielobjekts
- ▶ Spielwelt ist eine Menge von **Instanzen** der Spielobjekte

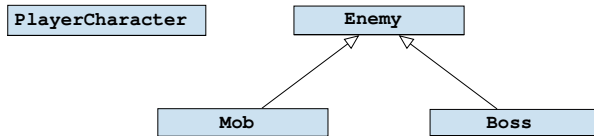
Eigenschaftszentriert

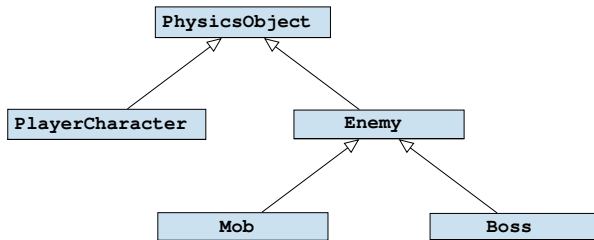
- ▶ Jedes Spielobjekt ist nur eine **ID**
- ▶ Tabellen für Eigenschaften aller Objekte
- ▶ Verhalten durch Operationen auf Tabellen
- ▶ Ähnlich zu Datenbanken

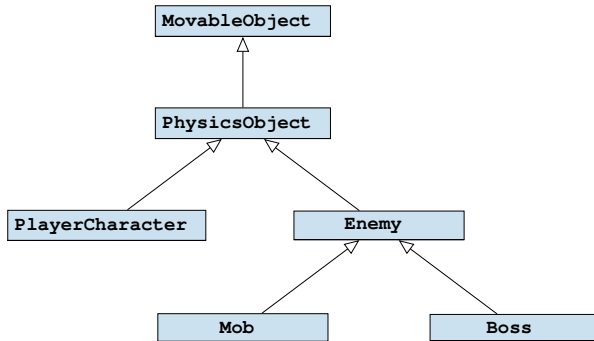
Mob

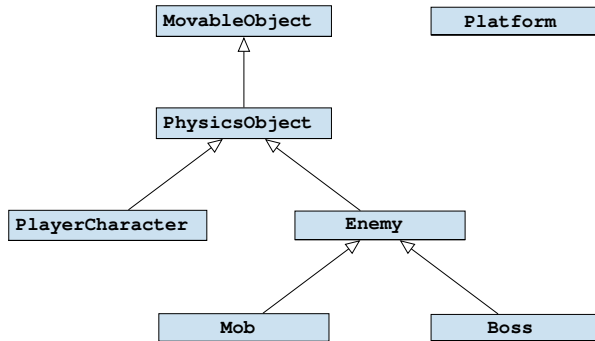
Boss

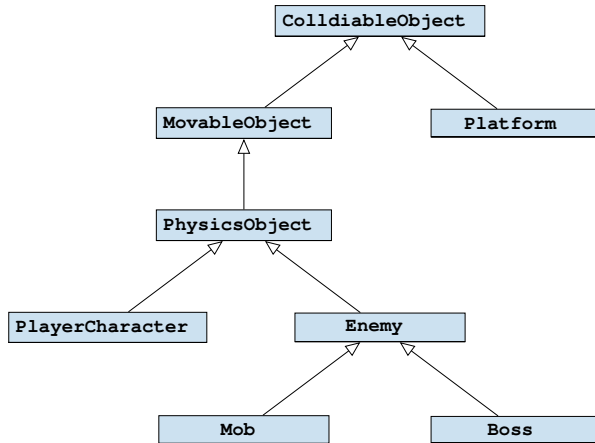


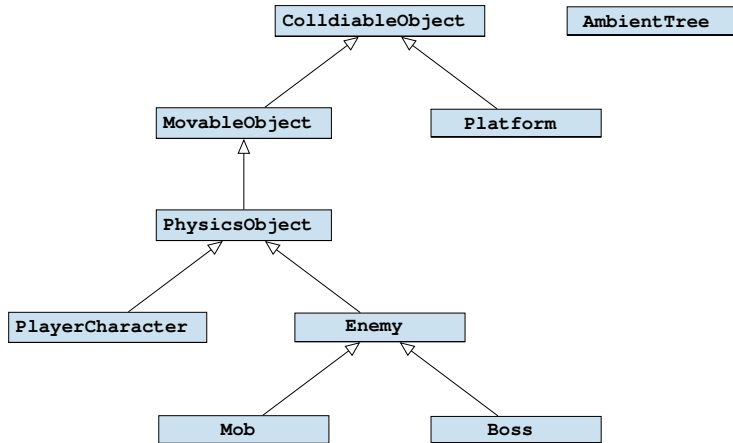


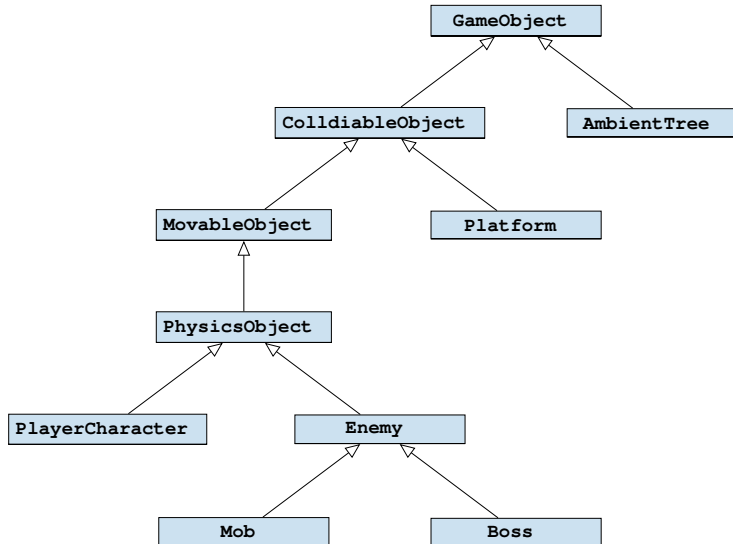


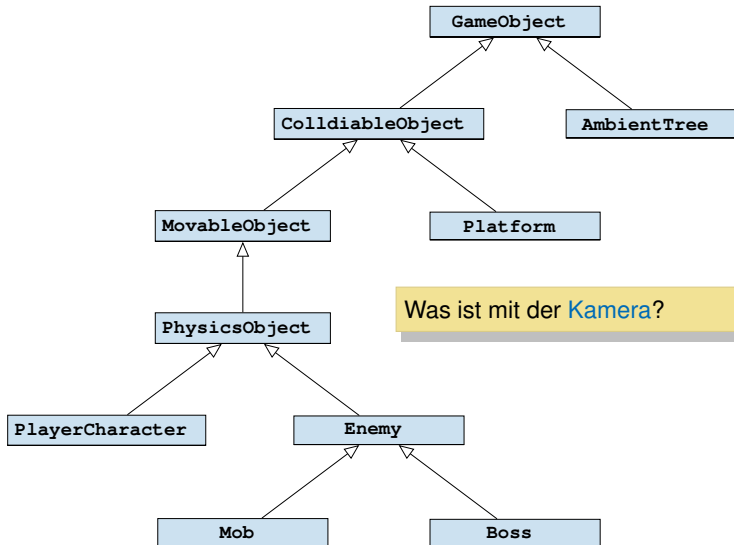






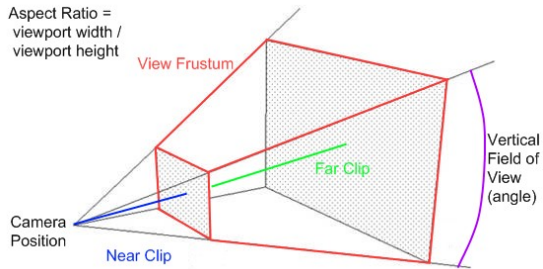






Eine einfache Kamera

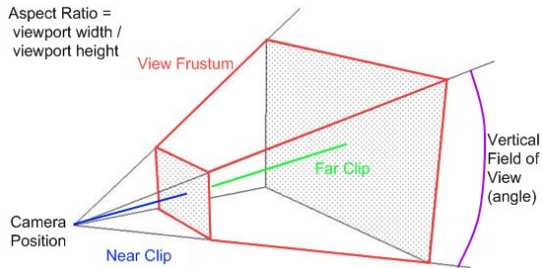
- ▶ Speichert z.B.:
 - ▶ View Matrix
 - ▶ Projection Matrix
- ▶ Grundlage für Picking
- ▶ Definiert **View Frustum**



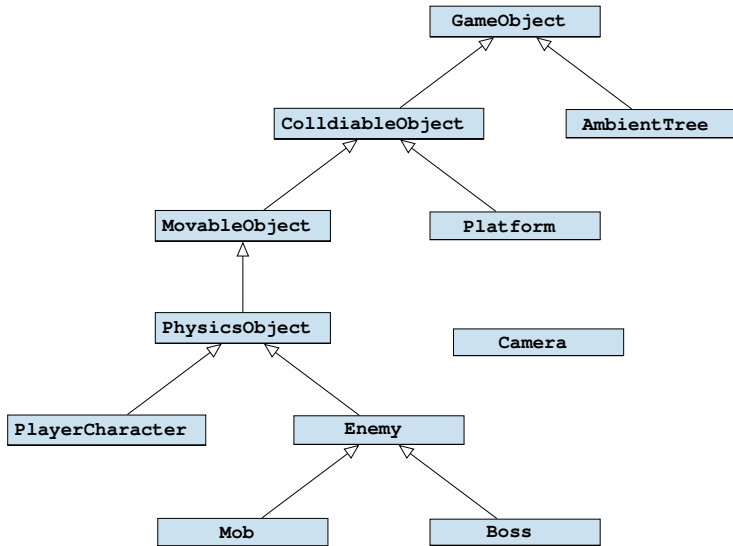
Eine einfache Kamera

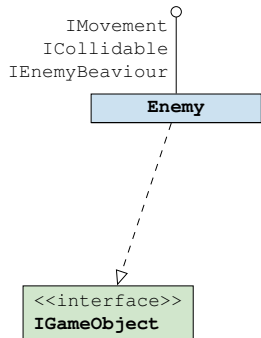
- ▶ Speichert z.B.:
 - ▶ View Matrix
 - ▶ Projection Matrix
- ▶ Grundlage für Picking
- ▶ Definiert **View Frustum**

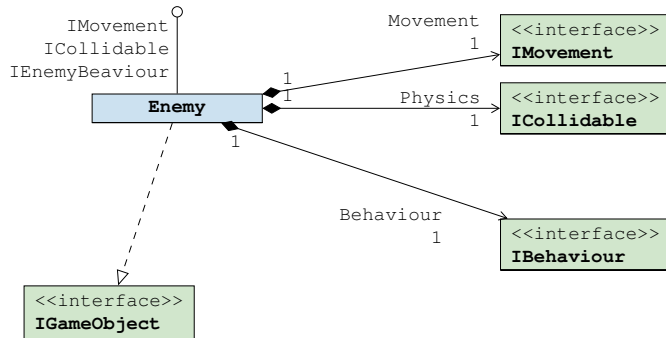
- Teil der Spielwelt
- Nicht **kollidierend**
- Beweglich

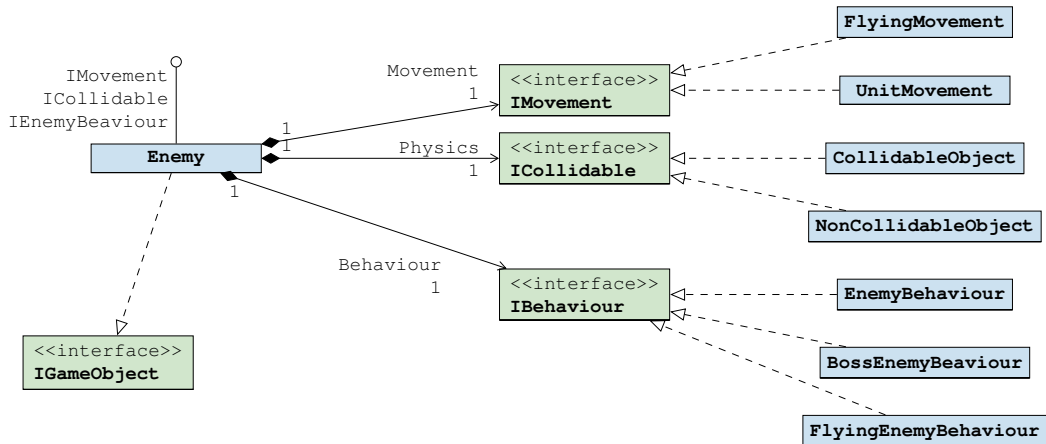


	Tree	Platform	PlayerCharacter	Mob	BossMob	Camera
GameObject	✓	✓	✓	✓	✓	✓
Collidable		✓	✓	✓	✓	
Movable			✓	✓	✓	✓
Physics			✓	✓	✓	
EnemyBehaviour				✓	✓	









Libraryfalle

Die Libraryfalle

- ▶ Es existieren viele Libraries
- ▶ Pro:
 - ▶ Rad nicht neu erfinden
 - ▶ Optimierte Lösungen für bekannte Probleme
- ▶ Aber:
 - ▶ Problem muss verstanden sein
 - ▶ Library muss verstanden werden
 - ▶ Versteckte Kosten wenn Library fehlerhaft/schlecht/unpassend/...

Die Libraryfalle

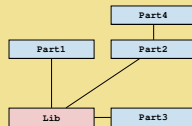
- ▶ Es existieren viele Libraries
- ▶ Pro:
 - ▶ Rad nicht neu erfinden
 - ▶ Optimierte Lösungen für bekannte Probleme
- ▶ Aber:
 - ▶ Problem muss verstanden sein
 - ▶ Library muss verstanden werden
 - ▶ Versteckte Kosten wenn Library fehlerhaft/schlecht/unpassend/...
- ▶ Vorher das Risiko abschätzen:
 - ▶ Wie zentral ist die Library?
 - ▶ Wie reif ist die Library?
 - ▶ Können wir das einfacher selbst machen?

Die Libraryfalle

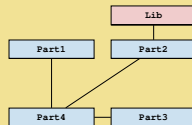
- ▶ Es existieren viele Libraries
- ▶ Pro:
 - ▶ Rad nicht neu erfinden
 - ▶ Optimierte Lösungen für bekannte Probleme
- ▶ Aber:
 - ▶ Problem muss verstanden sein
 - ▶ Library muss verstanden werden
 - ▶ **Versteckte Kosten** wenn Library fehlerhaft/schlecht/unpassend/...
- ▶ **Vorher** das Risiko abschätzen:
 - ▶ Wie zentral ist die Library?
 - ▶ Wie reif ist die Library?
 - ▶ Können wir das einfacher selbst machen?

Wie zentral ist die Library?

- Viel möglicher Schaden:



- Weniger möglicher Schaden:



Organisation

Ziel

Architektur anhand von Designprinzipien verbessern, Probleme frühzeitig aufdecken.

Ablauf:

- ▶ Kurze Zusammenfassung des Spiels
- ▶ Übersicht über die Architektur
 - ▶ Was machen die einzelnen Komponenten
- ▶ Vorstellen der Architektur anhand der Szenarien
- ▶ Fragen und zusätzliche Szenarien

Quelle: Dozent

Auslöser: Spieler

Umgebung: Ich befinde mich im Hauptmenü.
Ich drücke auf Spiel Starten, es werden
Spielobjekte erzeugt.

Ziel

Architektur anhand von Designprinzipien verbessern, Probleme frühzeitig aufdecken.

Ablauf:

- ▶ Kurze Zusammenfassung des Spiels
- ▶ Übersicht über die Architektur
 - ▶ Was machen die einzelnen Komponenten
- ▶ Vorstellen der Architektur anhand der Szenarien
- ▶ Fragen und zusätzliche Szenarien

Quelle: Dozent

Auslöser: Spieler

Umgebung: Ich befinde mich im Hauptmenü.
Ich drücke auf Spiel Starten, es werden
Spielobjekte erzeugt.

Von Anfang an Versuchen:

- Was ist Engine, was konkretes Spiel?
- Was gehört zusammen?
- Was hat nichts miteinander zu tun?
- Separation of Concerns

- ▶ Fangen Sie jetzt an zu programmieren
 - ▶ In **einem** Projekt
 - ▶ Arbeiten Sie gemeinsam
 - ▶ Versuchen Sie Ihr Spiel immer wieder zu spielen
- ▶ GDD-Reviews lesen und Probleme beheben
 - ▶ Inhaltliche Probleme früh lösen
- ▶ Nicht vergessen:
 - ▶ Architekturbesprechung (Doodles per Ticket)
 - ▶ Ideenpräsentation

- ▶ Fangen Sie jetzt an zu programmieren
 - ▶ In **einem** Projekt
 - ▶ Arbeiten Sie gemeinsam
 - ▶ Versuchen Sie Ihr Spiel immer wieder zu spielen
- ▶ GDD-Reviews lesen und Probleme beheben
 - ▶ Inhaltliche Probleme früh lösen
- ▶ Nicht vergessen:
 - ▶ Architekturbesprechung (Doodles per Ticket)
 - ▶ Ideenpräsentation

Sprechstunde:
Donnerstags 14-18 Uhr im Pool.

