# Projekt Abschluss Notizen, 1. Juni 2022:

Umgebung: Visual Studio 2022, C# 9, .Net 4.8

Ausführbar als Kommandozeilenanwendung

Sowohl die Abgabe- (Lehrerforderung) und private Version sind ausführbar, und mit obiger Umgebung bearbeitbar. Die .sln Projektdatein sind für VS in den jeweiligen Ordnern hinterlegt.

Start: 28. März 2022

Abgabetermin: 3. Juni 2022 (abgegeben am 30. Mai 2022)

## Lehreranforderungen: (wechselten über die Termine hinweg)

28. März:

Ziel: Schreibe eine objektorientierte Konsolen Applikation einer Ameisen- (Bienen) Simulation, in der ca. 100 Ameisen (Bienen) in ihrer Welt (Bildschirm) Futter suchen und zum Bau zurückbringen.

* Was für Entitäten (Datentypen) braucht man
* Lege für diese Datentypen direkt Klassen an
* Fange mit einer Ameise an, die sich bewegt (ein A, das von links nach rechts über den Monitor läuft)
* Nimm anschließend 10
* Ameisen sollen sich unabhängig voneinander bewegen

29.März (mündlich auf Nachfrage):

Ameisen oder Bienen (egal)

2D-Welt

80 \* 25 große Welt minimal (Größe des Kommandozeilenfensters)

Bis zu 100 Ameisen

Die simulierte Welt darf auch größer als 80 \* 25 sein, und man sieht nur den 80 \* 25 Ausschnitt, oder man vergrößert das Fenster

Eine Ameise kann z.B. andere Ameisen fressen, oder Futter suchen, oder Hindernisse umgehen

Etappenziele:

1. Eine Ameise darstellen
2. Ameise bewegen
3. Ameise für bestimmte Anzahl Zyklen bewegen lassen (zufällig)
4. 10 Ameisen implementieren
5. Kollisionsvermeidung oder Detektion und Aktion (Sterben, Weg-Drücken zum nächsten freien Feld, stehen bleiben…)
6. 100 Ameisen implementieren
7. Futter suchen, oder Ameisen fressen sich gegenseitig, oder Hindernisse umgehen
8. Futter zum Bau zurück bringen

Gedanken:

* Entitäten könnten auch größer als 1 \* 1 sein, aber dies würde Lernziel (OOP) verfehlen
  + Trotzdem Klassen so schreiben, dass Entitäten derart implementiert werden könnten?

25. April:

* Konkrete Anforderungen an die Simulation
* Anzahl der Tiere und Dauer der Simulation sollen einstellbar sein
* Ameisen/Bienen haben einen Haufen/Bau, in dem die Königin sitzt
* die Tiere streunen herum kommen aber regelmäßig (spätestens, wenn die Energie verbraucht ist) zum Bau zurück
* Tiere werden ungefähr 5 Tage alt
* Nicht Königinnen bewegen sich ca. einen Bildschirmschritt je Sekunde

2. Mai:

1. Konstruktor   
   Recherchiert, was ein Konstruktor ist und und wozu er gut ist. Formuliert eine kurze Erklärung von ein bis zwei Sätzen.
2. Die Ameisen werden "ungefähr" 20 Tage alt. Entwerft eine Lösung zur Bestimmung eines zufälligen Wertes "rund um 20. Baut dieses Verfahren so in den Konstrutor ein, dass die Anforderung erfüllt ist
3. Vererbung   
   Recherchiert, was es mit der Vererbung auf sich hat, wie diese im Code umgesetzt wird und nutzt diese zur Realisierung einer Klasse "Arbeiter-Ameise", die zusätzlich zu den bisherigen Ameisen auch eine Traglast hat. Implmentiert die Lösung in der Ameisen-Simulation.

Am 23. Mai haben wir eine Klausur zum Thema geschrieben.

## Meine Anforderungen:

* Grafik und Spielinhalt soll getrennt sein, damit der Spielinhalt als Spielstand in einer Datei gespeichert werden kann in späteren Programmversionen
* Starke Modularität und Erweiterbarkeit, damit ich später interessante Features einbauen kann.
  + Alle in der Welt vorhandenen „physischen Objekte“ sollen als abgeleitete Klasse von Entität aus implementiert werden.
    - Das heißt der Ameisenhügel ist eine Entität, und das Futter ist auch eine Entität; Die Arbeiterameise und die Königin sind jeweils von der einfachen Ameise abgeleitet.
* GitHub Nutzung, und Teilen meines Repositories mit Mitschülern, damit diese motiviert werden auch GitHub zu nutzen.
* Logging Datei wird erstellt und fortwährend zur Programmausführung beschrieben.

## Gedanken zur Lehrerversion:

1. Anforderungen wurden vom Lehrer sehr vage vorgegeben, sie änderten sich sogar zwischen zwei Wochen.
2. Auf Nachfrage konnte geklärt werden, dass auch eine sehr einfache Version ausreichend ist, sogar ohne Futter ist dies in Ordnung. Das geforderte Klassendiagramm würde sonst zu groß werden.
3. Als ich das abgespeckte Projekt abgeben wollte, kam es zu Copy-Paste-Fehlern. Alle Klassen in einer Datei unter zu bringen war sehr aufwendig. Die Copy-Paste-Fehler wurden sogar noch schlimmer, da ich die Einzeldatei nicht gespeichert hatte, sondern direkt bei Onenote in Teams im „Klassennotizbuch“ hinterlegt hatte. Der aus der App zurück kopierte Text hatte dann verdoppelte Zeilenumbrüche. Diese sollte ich dann nach Rücksprache mit dem Lehrer entfernen, da dies ja bei einer Abgabe beim Kunden auch stark bemängelt werden würde! Den folgenden Tipp dafür einen starken Editor zu nehmen, der „\n\n“ zu einem „\n“ ersetzt habe ich aber dankend befolgt.

## Gedanken zur privaten Version:

1. Es kam zu starken Problemen bei der Implementierung der „Food“ Klasse. Jene war mittels beidseitiger Assoziation implementiert. Dies war ein Fehler, ich habe versucht diese Implementierung umzugestalten. Die Arbeit an jener Beziehungsimplementation hat mir einige hilfreiche Erkenntnisse gegeben.
2. Ich habe erstmalig ein „Single Point of Truth“ Prinzip versucht zu implementieren.
3. Eine spätere Version hatte sogar eine DrawPower für einzelne Bildpunkte, damit z.B. das H für Hügel über einem @ der Ameise gezeichnet wird, und das H nie verdeckt ist

## Spätere Erweiterungsideen

* Innenleben der Kolonie
  + Größere Weltdimensionalität
* Statistiken
* Neue Königin durch Arbeiter erstellen
* Blattlausfarm
* Weg/Pheromonameisen
  + Legen Wegpunkte z.B. zu Ansammlungen von Nahrung
* Größere Nahrungsquellspots
* Hotkeys: f. Maxcount / Simulationsgeschwindigkeit ändern / weitere
* Kämpferameisen
* Schädlinge die Ameisen infizieren/Fressen (Hornissen, Grashüpfer o.ä.)
* Abwechselndes Anzeigen von z.B. „H“ und „Q“ falls sich zwei gleichrangige Objekte an einem Punkt befinden.
* Futter als tatsächliche Komposition.