

Entwurf

„The Sim“

Musterfirma GmbH & Co KG

| | | | | |
|---------------------------|--|----------------|---|----------------|
| Projektbezeichnung | „The Sim“ | | | |
| Projektgeber | Senckenberg Forschungsinstitut & Naturmuseum | | | |
| Projektleiter | Projekt Manager Daniel Czeschner | | | |
| Erstellt am | 09.05.2022 | | | |
| Letzte Änderung am | 25.05.2022 | | | |
| Status | | in Bearbeitung | X | fertiggestellt |
| Aktuelle Version | 3.0 | | | |

Änderungsverlauf

| Datum | Version | Geänderte Kapitel | Art der Änderung | Autor | Status |
|------------|---------|-------------------|--|------------------------------|--------|
| 09.05.2022 | 1.0 | Alle | Erstellung | K.G., D.C., R. K., L.S. | i. B. |
| 10.05.2022 | 1.1 | Alle > 1 | Erstellung / Anpassung | K.G., D.C., R.K., L.S. | i. B. |
| 14.05.2022 | 1.2 | Alle > 3 | Erstellung / Anpassung | K.G., D.C., R.K., E.S. | i. B. |
| 16.05.2022 | 1.3 | Alle >= 3 | Erstellung / Anpassung | K.G., D.C., R.K., E.S., L.S. | i. B. |
| 18.05.2022 | 2.0 | 4 bis 7 | Grundlegende Veränderung der Architektur | K.G., D.C., R.K., E.S. | i. B. |
| 19.05.2022 | 2.1 | 4, 5, 6, 7, 8 | Diagramme & kleine Anpassungen | D.C., E.S., K.G. | f. |
| 25.05.2022 | 3.0 | 5 | Überarbeitung | D.C. | f. |

Inhaltsverzeichnis

| | | |
|----|---|----|
| 1. | Allgemeines | 1 |
| | Glossar | 1 |
| | Team | 1 |
| 2. | Einleitung | 2 |
| 3. | Logische Sicht / Produktübersicht | 3 |
| | Schichten Strukturierung | 3 |
| | Identifizierte Module | 3 |
| 4. | Simulationsverhalten | 5 |
| | Allgemein | 5 |
| | Simulationsloop | 5 |
| | Rendering | 6 |
| | Dinosaurierverhalten | 6 |
| | Auswirkungen der Dinosauriereigenschaften | 10 |
| | Pflanzen (Obstacles) | 11 |
| 5. | Schnittstellen | 13 |
| 6. | Dinosaurierzustände und Transitionen | 14 |
| | Pflanzenfresser | 14 |
| | Fleischfresser | 15 |
| | Allesfresser | 16 |
| 7. | Datenhaltung/Datenverarbeitung | 17 |
| | Simulationsobjekt-Konfiguration | 17 |
| | Szenariokonfiguration | 18 |
| | Landschaftkonfiguration | 18 |
| | Statistiken | 18 |
| 8. | Simulationsloop | 19 |
| | Automatisch | 19 |
| | Schrittverfahren | 20 |
| 9. | GUI | 21 |

Abbildungsverzeichnis

| | |
|--|----|
| Abb. 1: Komponentendiagramm..... | 4 |
| Abb. 2: Klassendiagramm..... | 13 |
| Abb. 3: Zustandsdiagramm Pflanzenfresser | 14 |
| Abb. 4: Zustandsdiagramm Fleischfresser | 15 |
| Abb. 5: Sequenzdiagramm-Simulationsloop-auto | 19 |
| Abb. 6: Sequenzdiagramm-Simulationsloop-step..... | 20 |
| Abb. 7: Mockup - Simulationsoberfläche..... | 21 |
| Abb. 8: Mockup - Konfiguration | 21 |
| Abb. 9: Mockup - Statistiken | 22 |

Tabellenverzeichnis

| | |
|--------------------------------|---|
| Tab. 1: Glossar | 1 |
| Tab. 2: Teammitglieder | 1 |
| Tab. 3: Schichtentabelle | 3 |

1. Allgemeines

Glossar

| Bezeichnung | Bedeutung |
|----------------------|--|
| <i>The Sim</i> | Bezeichnung der beschriebenen Anwendung, welche dieses Dokument beschreibt |
| <i>Kunde</i> | Senckenberg Forschungsinstitut und Naturmuseum Frankfurt |
| <i>Projektgeber</i> | |
| <i>Zielgruppe</i> | Besucher des Kunden |
| <i>Projektnehmer</i> | Musterfirma GmbH & Co. KG |

Tab. 1: Glossar

Team

| Rolle(n) | Name | E-Mail |
|---|---------------------|----------------------------------|
| Projekt Manager, Softwareengineer | Daniel Czeschner | s200285@student.dhbw-mannheim.de |
| Communication Manager, Softwareengineer | Kai Grübener | s200291@student.dhbw-mannheim.de |
| Softwareengineer | Eric Stefan | s200305@student.dhbw-mannheim.de |
| Softwareengineer | Lucas Schaffer | s200301@student.dhbw-mannheim.de |
| Softwareengineer | Robin Khatri Chetri | s200296@student.dhbw-mannheim.de |
| Softwareengineer | Tamina Mühlenberg | s200299@student.dhbw-mannheim.de |

Tab. 2: Teammitglieder

2. Einleitung

Dieses Dokument dient dem internen Entwurf für das Projekt "The Sim" des Kunden *Senckenberg Forschungsinstitut und Naturmuseum Frankfurt*. Die Funktionen und Anforderungen an die Software sind der aktuellen Version des Pflichtenheftes zu entnehmen.

Zweck dieses Dokumentes ist es, den Entwurf der Software für die spätere Umsetzung festzuhalten, sowie die Gedankengänge und Entscheidungen beim Design zu dokumentieren, um als ein Teil der „Entwickler-Dokumentation“ für spätere Anpassungen zu fungieren.

3. Logische Sicht / Produktübersicht

Schichten Strukturierung

Um die Anforderungen aus dem Pflichtenheft umsetzen zu können, wird eine Software benötigt, welche verschiedene Funktionen bzw. Funktions-Kategorien besteht. Es liegt nahe als erste Strukturierung, die Funktionen in Schichten zu unterteilen.

Das Produkt kann in folgende logische Schichten aufgeteilt werden:

| | | |
|--|-------------|-----------------------|
| Benutzeroberfläche (GUI) | | |
| Simulationsansicht | Statistiken | Konfigurationsansicht |
| Simulationslogik (später weiter zu spezifizieren) | | |
| Dateisystem & Treiber | | |

Tab. 3: Schichtentabelle

Die Anordnung der Schichten gibt hierbei Aufschluss über die „Abstraktions-Ebenen“. Während das Dateisystem (Konfigurationsdateien; siehe Kapitel 7) etwas näher an der Hardware angesiedelt ist, stellt die GUI eine abstrakte Ebene dar, welche dem Benutzer die Bedienung der Software ermöglicht, ohne nähere Details zur Umsetzung zu kennen.

Identifizierte Module

Von den obigen Schichten lässt sich das Projekt in folgende, zu bearbeitende, Module konkretisieren:

- Konfiguration
- Import/Export
- Simulation (Core)
- Statistiken
- GUI durch JavaFX

In welchem Zusammenhang die Module zueinanderstehen und in welche Richtung die Kommunikation zwischen ihnen verläuft, wird in folgendem Diagramm veranschaulicht:

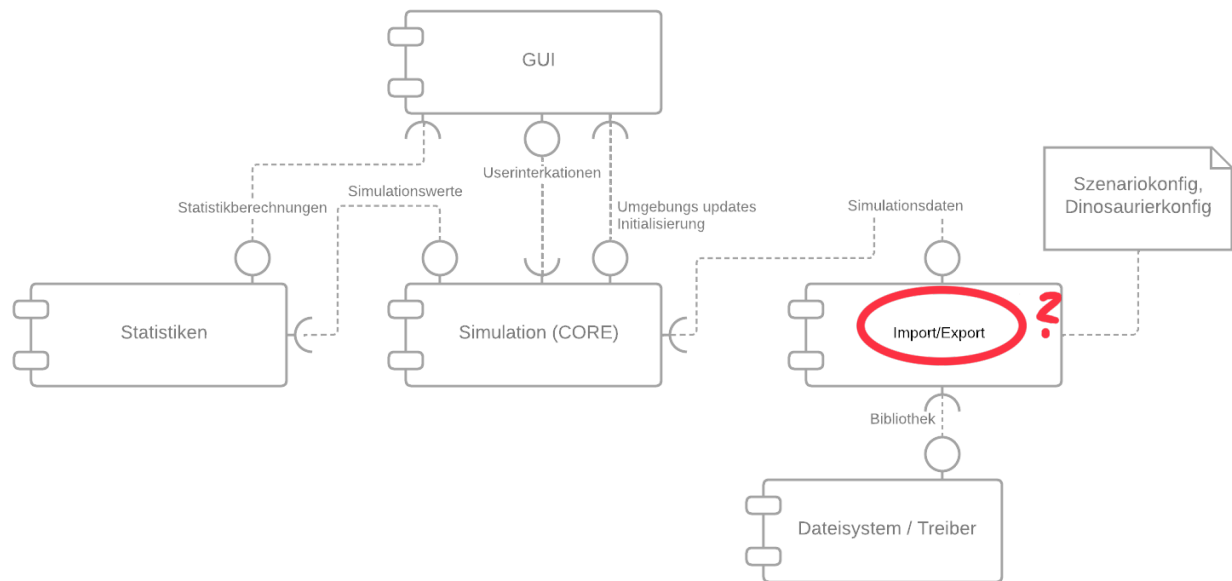


Abb. 1: Komponentendiagramm

Als Modul, welches direkt mit dem Nutzer interagiert, wurde das GUI-Modul ausgewählt, um die zentrale Koordination zwischen den anderen Modulen vorzunehmen. Diese Entscheidung verspricht den Datenfluss möglichst übersichtlich zu gestalten, indem Schnittstellen zwischen den Modulen minimiert werden.

4. Simulationsverhalten

Zunächst beschreiben wir das Simulationsverhalten schriftlich, um einen besseren Überblick zu geben und somit die UML-Diagramme im Anschluss besser verstehen zu können.

Allgemein

Zu Beginn der Simulation, werden alle Simulationsobjekte erzeugt, dazu gehören die Dinosaurier und Pflanzen. Dabei werden die Werte der Eigenschaften von diesen unter Berücksichtigung der im Pflichtenheft erwähnten Varianz [?] festgelegt, wodurch einzelne Dinosaurier und Pflanzen gleicher Art trotzdem unterschiedliche Werte haben können (Der in ihrer Konfiguration angegebenen Varianz folgend).

Damit die 2D-Simulation grafisch dargestellt werden kann, wird eine Simulationsoberfläche benötigt. Um dort die Welt aus verschiedenen Landflächen zusammensetzen zu können sowie gezielt Landflächen, welche nicht von bestimmten Dinosauriern durchquert werden können, auszuweichen zu können, wird etwas zur Einteilung dieser Oberfläche benötigt. In diesem Zusammenhang bietet sich dabei ein GridSystem an, welches jedoch ausschließlich für den Hintergrund gilt. Dieses besteht aus einem 2-Dimensionalen-Array, welches Tile-Objekte enthält, die zum einen das Bild dieses Tiles besitzen und zum anderen Variablen, die aussagen, ob ein Dinosaurier darüber laufen kann oder nicht (siehe Kapitel 5: „Schnittstellen“). Im GridSystem wird daneben noch die Angabe benötigt, wie groß (hoch und breit) ein Gridfeld ist. Dies entspricht der Größe von einem Tile/Gridfeld welches 45x45 Pixel groß ist. Dadurch ergibt sich, unter Berücksichtigung der Simulationcontrol-Sidebar, etwa eine Gridgröße von 36x24 Feldern bei Full-HD.

Um die Simulationsobjekte sowie den Untergrund darstellen zu können, muss die Benutzeroberfläche dies ermöglichen. Hierfür können für den Untergrund ein Canvas (siehe Kapitel Rendering) und für die Simulationsobjekte JavaFX-Objekte (ImageViews) verwendet werden, um diese an ihren Positionen darzustellen. Damit man nun auch weiß, welche Position ein Simulationsobjekt hat, wird eine Positionsangabe benötigt. Da wir uns in einer 2D-Simulation befinden, benötigen wir also eine X- sowie Y-Koordinate, welche wir als Vector2D bezeichnet haben (Ursprung des Koordinatensystems ist oben links). Jedes Objekt hat daher genauso einen Vektor für seine Position auf der Oberfläche, wodurch Berechnungen wie die Entfernung zwischen zwei Simulationsobjekten ermöglicht werden. Beim Erzeugen der Simulationsobjekte wird die Position zufällig aus freien Gridfeldern gewählt und mittig in diesem platziert.

Simulationsloop

Die einzelnen Verhaltensweisen der Simulation werden je nach zuvor gewählter Simulations-Geschwindigkeit wiederholt hintereinander aufgerufen. Die Simulationsloop ist hierbei dafür verantwortlich die Status der Simulationsobjekte zu aktualisieren. Dies ist dabei abhängig von ihrem Verhalten/State (Behavior Trees). Abhängig von ihrem aktuellen State werden Werte wie z.B. die Position geändert. [?] Nachdem alle

Simulationsobjekte ihren Simulationsdurchlauf haben, wird die Darstellung, bspw. ihre Position, in JavaFX aktualisiert. Im Simulationsschrittverfahren wird anders wie bei der Simulationloop erst nach einer Anzahl von 30 Updates multipliziert mit der Schrittweite ein aktualisieren der Darstellung getriggert. Damit unter anderem Bewegungen im automatischen Verfahren (Simulationloop) flüssig aussehen, muss die Updaterate höher als einmal die Sekunde sein. Ein guter Wert ist es mindestens 60x die Sekunde ein Update zu triggern.

→ Einheit?

Damit, falls die Updaterate von 60 im automatischen Verfahren nicht erreicht werden kann, nicht die Dinosaurier unter anderem langsamer laufen, da weniger Updates getriggert werden, wird eine Delta-Updatezeit benötigt. Diese wird unter anderem bei den Updates der Positionen durch eine Bewegung eines Dinosauriers multipliziert. Bei dem Simulationsschrittverfahren ist dabei zu berücksichtigen, dass es keine wirkliche Updaterate und somit keine Delta-Updatezeit. Hier entspricht die Delta-Updatezeit etwa dem Wert 0,1. Dieser Wert wurde durch einen Prototyp der Software als angemessen empfunden.

Rendering

Für die Darstellung der Simulation, wird die simulierte Umgebung mithilfe einer Rastergrafik (Grid) innerhalb eines Canvas-Elementes visualisiert. Die Simulationsmap muss dafür nur zu Beginn der Simulation einmal mit der gewählten Landschaft gerendert werden. Hierfür werden die Grafiken (Sprites) der einzelnen Tiles an ihren entsprechenden Positionen im Canvas dargestellt (Grid position to canvas position). Die Simulationsobjekte müssen dabei nicht von uns gerendert werden, da das Rendern und Neurendern nach Positionsupdates dieser Objekte JavaFX übernimmt, da es sich bei diesen um Java-FX-Objekte handelt. Lediglich die Positionsänderungen müssen durch uns getriggert werden.

Dinosaurierverhalten

Das Verhalten von Dinosauriern muss innerhalb der Simulation ebenfalls realisiert werden. Um die Verhaltensweisen möglichst realistisch abbilden zu können, haben Dinosaurier eine begrenzte radiale Sichtweite (je Art unterschiedlich) sowie eine maximale Interaktionsweite (je Art unterschiedlich). Die Interaktionsweite besagt, wie weit ein Dinosaurier maximal entfernt sein kann, damit eine Interaktion (wie bspw. trinken, essen, ...) stattfinden kann. Wie in der aktuellen Version des Pflichtenheftes beschrieben, müssen Dinosaurier Nahrung und Flüssigkeit zu sich nehmen (F-SYS-40.2). Diese werden pro Dinosaurier als "Füllstand" in Ganzen Zahlen angegeben, um Nahrung und Hydration besser abbilden zu können. Liegen beide dieser Werte über 50% so bewegt sich der Dinosaurier frei herum oder beschäftigt sich mit einen der später beschriebenen Verhaltensmustern.

Liegt mindestens einer dieser Werte unter 50%, so begibt sich der Saurier auf die Nahrungs- / Flüssigkeitssuche. Liegen beide Werte darunter, priorisiert er entsprechend der Dringlichkeit und Distanz zur Quelle.

Dinosaurier haben insgesamt verschiedene Zustände / Aktionen die sie ausüben können.

Diese Aktionen von Dinosauriern benötigen Zeit. Bis auf die Bewegung kann hierbei davon ausgegangen werden, dass eine Aktion 120 Updates benötigt (also das doppelte der Updaterate). Im Folgenden werden die Bewegungen näher erläutert und vor allem wie entschieden wird, ob ein Dinosaurier zu einem Objekt oder Tile laufen kann oder nicht.

Wandern

Im Wandern-Zustand bewegt sich ein Dinosaurier zufällig zu einer Position in seiner Sichtweite. Dabei wird ein Punkt gewählt, welcher mindestens 50% der Sichtweite von seinem Ursprung entfernt liegt. Anschließend muss geprüft werden, ob sich dieser Punkt in einem Tile befindet, welcher nicht von dem Dinosaurier durchquert werden kann. Hierfür kann einfach die Position des Punktes auf ein Gridfeld zurückgerechnet werden. Liegt der Punkt auf einem Tile, welches der Dinosaurier nicht betreten kann, wird ein anderer Punkt gewählt. Falls er auf einem „gültigen“ Tile liegt, muss geprüft werden ob dieser Punkt in einer Interaktionsweite einer Pflanze oder eines Dinosauriers liegt. Ist dies nicht der Fall, kann der Dinosaurier zu diesem Punkt laufen, ansonsten muss ein anderer Punkt gewählt werden.

Nahrungsquelle suchen und sich zu dieser bewegen

Wenn sich ein Dinosaurier zu einer Nahrungsquelle bewegt bzw. auf Nahrungssuche ist oder einen anderen Dinosaurier jagt, verändern sich etwas die Überprüfungen.

Zuerst überprüft ein Dinosaurier anhand seiner radialen Sichtweite, welche anderen Simulationsobjekte innerhalb dieser liegen. Hierfür wird geprüft, ob die eigene radiale Sichtweite sich mit der Interaktionsweite aller anderen Pflanzen oder Dinosaurier schneiden (Sowohl Pflanzen als auch Dinosaurier haben eine Interaktionsweite, die logisch gesehen der Breite dieser entspricht). Wenn mehrere Pflanzen und/ oder Dinosaurier gefunden werden, werden diese entsprechend der Angaben in diesem Dokument (siehe Unterkapitel: „Auswirkungen der Dinosauriereigenschaften“) priorisiert.

Bevor das vorläufig priorisierte Simulationsobjekt endgültig als Ziel gewählt wird, wird eine logische „Line of Sight“ zu diesem Objekt gezogen (konkret eine Lineare Gleichung). Diese dient dazu um zu prüfen, ob innerhalb dieser (also dem **direkten** Weg) ein Hindernis auf dem Weg liegt. Ist dies der Fall, so sieht der Dinosaurier dieses Objekt gar nicht erst und wählt das zweit-priorisierte Simulationsobjekt als vorläufiges Ziel. Nun wird erneut eine „Line of Sight“ gespannt und auf Hindernisse geprüft, ... usw.

Wenn im Endeffekt kein Simulationsobjekt mehr als Ziel übrigbleibt, so bleibt der derzeitige Zustand des Dinosauriers erhalten.

Nach dem Prüfen auf Hindernisse ist es außerdem noch erforderlich alle Grid-Felder die sich auf direktem Weg zum gewählten vorläufigen Ziel befinden darauf zu prüfen, ob diese durch den suchenden Dinosaurier überhaupt durchquert werden können. Bspw. wenn ein Grid-Feld Wasser enthält muss der Dinosaurier schwimmen können. Kann der Dinosaurier eines der Felder nicht durchqueren, so wird das nächst-priorisierte Ziel als neues vorläufiges Ziel gewählt ... usw. Hierfür bietet sich der Bresenham Line Algorithmus an. Es handelt sich dabei um einen gridbasierten Algorithmus, welcher versucht zwischen zwei Gridfeldern eine gerade Linie zu ziehen. Dabei gibt er alle Gridfelder zurück, die er schneidet, wodurch diese gezielt überprüft werden können.

Wenn dabei kein Simulationsobjekt mehr als Ziel übrigbleibt, so bleibt der derzeitige Zustand des Dinosauriers erhalten.

Wenn eine Pflanze als endgültiges Ziel gewählt wird, so wird diese gegessen, sobald sich die Interaktionsweite des Dinosauriers mit der Interaktionsweite der Pflanze schneidet. Die Pflanze wird dabei lokal statisch aufgenommen. Nach dem Verbrauch der Nahrungsquelle, wird diese als gefressen markiert, kann jedoch nachwachsen (Abhängig von der globalen Pflanzenwachstumsrate -> Objekt wird nicht gelöscht).

Wenn ein Fleisch- oder Allesfresser-Dinosaurier einen anderen Dinosaurier endgültig als Ziel wählt, so setzt sich dessen Attribut "target" entsprechend und auch der gejagte Dinosaurier bekommt ein Attribut-Update von "isChased". Während ein "target" gesetzt ist werden keine weiteren Überprüfungen der radialen Sichtweite und möglicher weiterer Ziele durchgeführt, sondern es werden lediglich die beiden "zusammengehörenden" Dinosaurier auf Sichtweite und Interaktionsweite gegeneinander geprüft. Sodass festgestellt werden kann, wann ein Dinosaurier entkommen konnte und wann dieser eingeholt und gefressen wird.

Eingeholt und gefressen werden kann ein gejagter Dinosaurier sobald sich die Interaktionsweiten des Jägers und des Gejagten schneiden. Wenn die Jagd also erfolgreich war also der gejagte Dinosaurier erlegt wurde, wird dieser von der Welt entfernt und nicht mehr angezeigt (Objekt wird gelöscht).

Dadurch, dass während des Jagens keine umfassenden Überprüfungen auf neue Ziele und damit auf Hindernisse durchgeführt werden, können Dinosaurier durch andere Dinosaurier einfach durchgehen auf dem Weg zu ihrem Ziel.

Fliehen

Wenn ein Dinosaurier gejagt wird, so versucht dieser zu fliehen. Dies geschieht dadurch, dass in dem Halbkreis seiner radialen Sichtweite, der am weitesten weg von dem Jäger ist ein zufälliger Punkt gewählt wird, zu dem der Gejagte flieht. Der zufällig gewählte Punkt muss dabei

mehr als 75% der maximalen radialen Sichtweite betragen. Ansonsten verhält sich das Fliehen ähnlich wie im oben beschriebenen Wandern-Zustand.

Logischerweise müssen hierbei die gleichen Überprüfungen auf Hindernisse und begehbaren Untergrund gemacht werden wie bei der oben beschriebenen Nahrungssuche.

Wird kein einziger möglicher Zielpunkt auf der Flucht gefunden, so bleibt der Dinosaurier stehen.

Backuplösung für die Bewegungen

Falls dies nicht so funktioniert wie geplant (aus Performance Gründen oder ähnlichem), gibt es als Backup-Lösung das Verfahren die Bewegungen gridbasiert zu machen. Dadurch wird in jedem Tile zusätzlich gespeichert, welche Simulationsobjekte sich in diesen befinden. Ist es besetzt kann ein Dinosaurier sich nicht auf dieses Feld bewegen. Die Interaktionsreichweite muss in diesem Fall entsprechend angepasst werden, dass diese ein größer ist als ein Gridfeld. Die Sichtweite beträgt in diesem Fall die überlappenden Gridfelder. Befindet sich was in diesen, kann es als Target gewählt werden.

Fortpflanzung

Ein weiteres komplizierteres Verhalten eines Dinosauriers stellt die Fortpflanzung dar. Sobald die Fortpflanzungswilligkeit eines Dinosauriers den Maximalwert erreicht, ist dieser dazu bereit sich mit einem weiteren Dinosaurier derselben Spezies zu paaren. Hierzu müssen sich zwei paarungsbereite (also beide haben maximale Fortpflanzungswilligkeit) Dinosaurier unterschiedlichen Geschlechts aber der gleichen Dinosaurierart innerhalb des Interaktionsradius befinden. Darauf folgend verweilen sie für die oben erwähnte allgemeine Aktionsdauer (120Updates) und es entsteht unmittelbar ein weiterer Dinosaurier derselben Spezies, welcher direkt aus einem Ei schlüpft. Der eigentliche Fortpflanzungsprozess wird gemäß den Anforderungen nicht dargestellt (F-GUI-30.8), ein dadurch gezeugter Dinosaurier erscheint auf einem freien Gridfeld neben seinen Eltern. Nach dessen Geburt besteht keine weitere familiäre Bindung und jeder Dinosaurier lebt für sich weiter. Der Wert des Fortpflanzungswillens, der bestimmt wie schnell ein Dinosaurier den Status der Paarungsbereitschaft erreicht, wird dabei abhängig von der jeweiligen Rasse festgelegt. Im Rahmen der Fortpflanzung erbt das Kind alle Eigenschaften der Eltern und zu 50% den Eigenschaftswert des Vaters und zu 50% den Eigenschaftswert der Mutter. Dabei besteht eine 20-prozentige Chance, bei jeder einzelnen Eigenschaft, auf eine Mutation, wobei Eigenschaften zufällig um bis zu 30% besser oder schlechter werden (z.B. Stärke).

Zustände / State Machine:

Das gesamte Verhalten eines Dinosauriers lässt sich somit in mehrere Zustände (States) zusammenfassen und durch eine State Machine oder einen Behavior Tree realisieren. Diese sind „sitzen“ (wartend),

„wandernd“ (sich zufällig bewegen), „Bewegung zur Nahrungsquelle / Wasserstelle“, „jagen“ (je nach Art), „fliehen“ (je nach Art), „fressen“ bzw. „trinken“ oder auch einfach „Nahrungsaufnahme“, „tot“ und „paaren“. Initialisiert wird ein Dinosaurier im wartenden Zustand und wechselt nach einer zufälligen Zeit z.B. 1 Simulationsloop seinen Zustand. Der Zustand ist auch abhängig von seinen Lebensnotwendigen Eigenschaften Nahrung und Hydration. Ist Nahrung und Hydration im definierten „Normalbereich“ wechselt er in den Zustand wandern und läuft zu einer zufälligen Position. Dort angekommen, wechselt er wieder in den Zustand wartend. Ist ein Wert der Lebensnotwendigen Eigenschaften bei einem Fleischfresser nach der erwähnten Gewichtung „kritisch“ wechselt er in den Zustand „jagend“ bzw. „Bewegen zur Nahrungsquelle“, abhängig davon, ob ein Pflanzenfresser oder eine Wasserquelle in Reichweite bzw. Line of Sight ist. Bei „jagen“ wird dem gejagten Dinosaurier mitgeteilt, dass er gejagt wird und versucht dem Dinosaurier zu entkommen (wechselt in den Zustand „fliehen“), indem er in eine andere Richtung läuft. Falls keine Beute in der Nähe ist, wechselt er in den Zustand „wandernd“. Bei Pflanzenfressern ist dies ähnlich, nur dass diese dann in den Zustand „Bewegung zur Nahrungsquelle“ wechseln, wenn eine Pflanze bzw. Wasserquelle in Reichweite (bzw. Line of Sight) ist, ansonsten wechselt er in den Zustand „wandernd“. Wenn die Jagd beziehungsweise das Fliehen erfolgreich war, also der gejagte Dinosaurier erlegt wurde oder entkommen konnte (aus dem Sichtfeld des Jägers verschwunden ist, gejagter Dinosaurier wird darüber informiert), wechseln die Dinosaurier in den Zustand „fressen“ beziehungsweise „tot“ bei erfolgreicher Jagd oder „wandernd“ beim Entkommen bzw. misslungener Jagd. Der genaue Ablauf kann den entsprechenden Diagrammen entnommen werden (siehe Kapitel 6 Dinosaurierzustände und Transitionen).

Auswirkungen der Dinosauriereigenschaften

Die Dinosauriereigenschaften haben Auswirkungen auf die Simulation. Diese werden im Folgenden näher erläutert:

-Nahrung

Der Wert „Nahrung“ startet bei dem individuellen Maximalwert und nimmt mit der Zeit ab. Wenn dieser Wert unter 50% fällt, wird der Dino sich auf Nahrungssuche begeben, also Beutetiere jagen bzw. Pflanzen suchen. Nach der Nahrungsaufnahme setzt sich der Nahrungswert auf 100% zurück. Findet der Dino keine Nahrung bevor der Nahrungswert 0% erreicht, verhungert dieser.

-Hydration

Der Wert „Hydration“ startet bei dem individuellen Maximalwert und nimmt mit der Zeit ab. Wenn dieser Wert unter 50% fällt, wird der Dino nach einem Gewässer in der Nähe suchen und trinken. Danach setzt sich der Hydrationswert auf 100% zurück. Findet der Dino kein Gewässer bevor der Hydrationswert 0% erreicht, verdurstet dieser.

-Stärke

Jagende Tiere benötigen genug Stärke um schwächere Tiere erlegen zu können.

-Geschwindigkeit

Je höher der Geschwindigkeitswert, desto schneller kann sich der Dino bewegen. Die Geschwindigkeit eines Dinosauriers gibt an, wie viele Pixel dieser innerhalb eines Simulationsloops (eines Updates) laufen kann.

-Fortpflanzungswilligkeit

Ein fester, aber individueller Wert der entscheidet wie schnell der Fortpflanzungswille steigt.

-Fortpflanzungswille

Der Wert „Fortpflanzungswille“ startet bei 0% und steigt mit der Zeit an, wobei das Maximum 100% beträgt. Zwei Dinos pflanzen sich nur fort, falls beide einen Fortpflanzungswillen von 100% besitzen und sich ihre Interaktionsweiten schneiden. Nach dem Paarungsprozess setzt sich der Fortpflanzungswillenswert auf 0% zurück.

-Gewicht

Ein numerischer Wert, welcher sich nicht weiter auf das Verhalten auswirkt.

-Größe

Ein numerischer Wert, welcher sich nicht weiter auf das Verhalten auswirkt.

-Kann schwimmen

Dieser Wahrheitswert entscheidet darüber ob sich der Dino im Wasser fortbewegen kann.

-Kann klettern

Dieser Wahrheitswert entscheidet darüber ob sich der Dino über Berge hinweg fortbewegen kann.

-Nahrungsart

Diese Eigenschaft entscheidet darüber welche Nahrungsmittel der Dino bei der Suche in Erwägung zieht. Die möglichen Werte sind vorab definiert für Pflanzen bzw. andere Lebewesen. Allesfresser präferieren (bei gleichzeitigem Vorkommen) die Zunahme pflanzlicher Nahrung, um die Anstrengung einer Jagd zu umgehen.

Pflanzen (Obstacles)

Pflanzen besitzen eine Wachstumsrate (gültig für alle Pflanzen) und eine Wuchsgröße von 100. Ein Dinosaurier kann eine Pflanze nur fressen, wenn diese ihre vollständige Wuchsgröße erreicht hat. Nach dem

Fressen der Pflanze wird dessen Wuchsgröße auf 0 gesetzt und wächst entsprechend der Wachstumsrate wieder nach.

Hindernisse wie Berge, Bäume oder Wasser besitzen kein Wachstum und verändern sich somit nicht im Laufe der Simulation. Alle Felder, welche durch Pflanzen oder Hindernisse belegt sind, können nicht von Dinosauriern betreten werden (Ausnahme: Kletterfähige Dinosaurier können Berge besteigen, schwimmfähige Dinosaurier können Wasser durchqueren).

5. Schnittstellen

Um die konkreten Anforderungen des Pflichtenhefts umzusetzen, lässt sich obige Beschreibung des Simulationsverhaltens unter Berücksichtigung des Moduldiagramms zu einem Klassendiagramm erweitern, indem die einzelnen Module und deren Interaktion detaillierter beleuchtet werden.

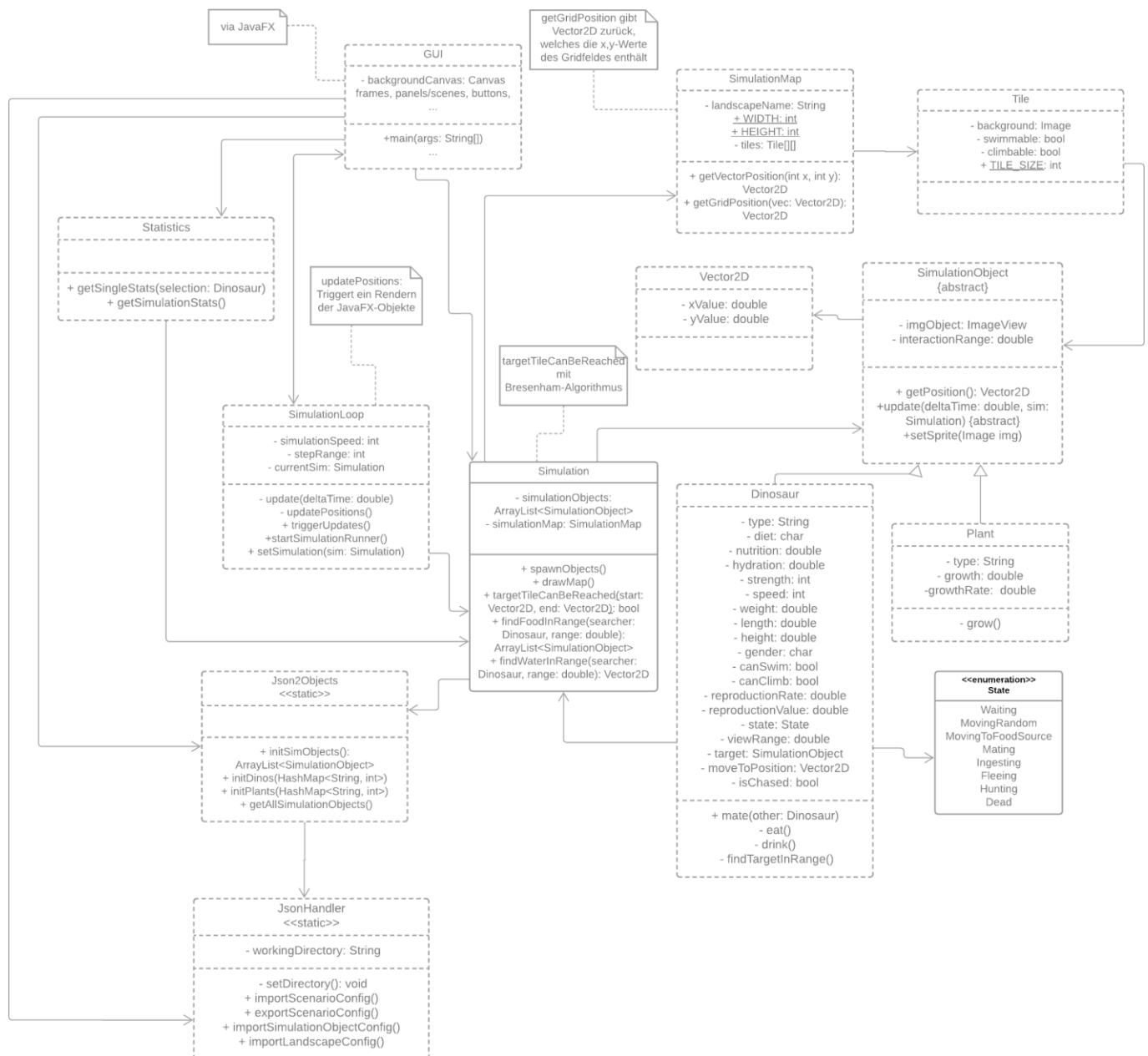


Abb. 2: Klassendiagramm



6. Dinosaurierzustände und Transitionen

Im Nachfolgenden werden die Dinosaurierzustände dargestellt sowie die einzelnen Transitionen zwischen diesen Zuständen. Dies wurden bereits im Kapitel 4 unter „Dinosaurierverhalten“ beschrieben und werden hier nun in Zustandsdiagrammen veranschaulicht.

Pflanzenfresser

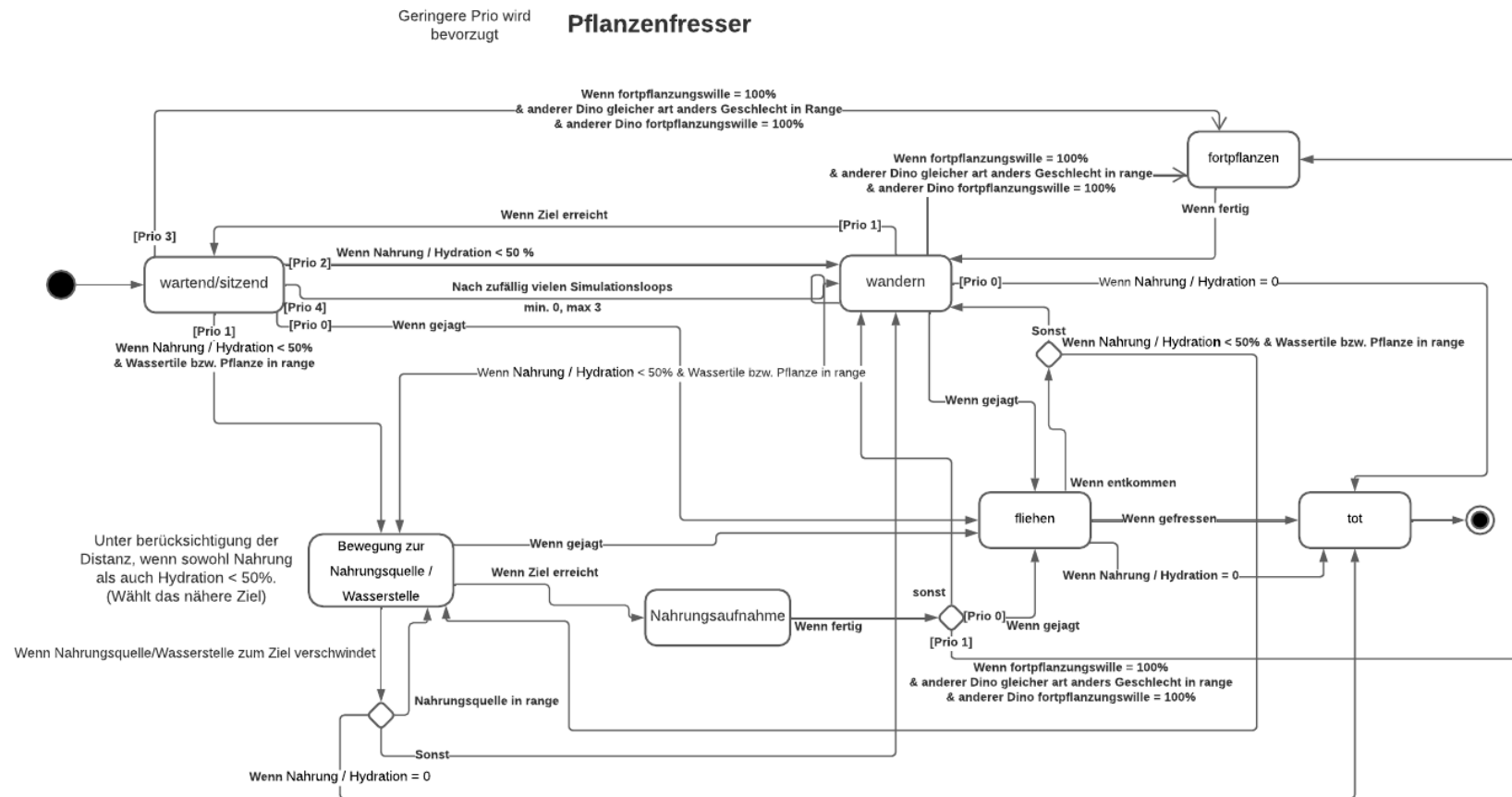


Abb. 3: Zustandsdiagramm Pflanzenfresser

Fleischfresser

Nachfolgend werden die Zustände eines Fleischfressers dargestellt. Dinosaurier die von einem Fleischfresser gefressen werden können, dürfen nicht von der gleichen Art sein.

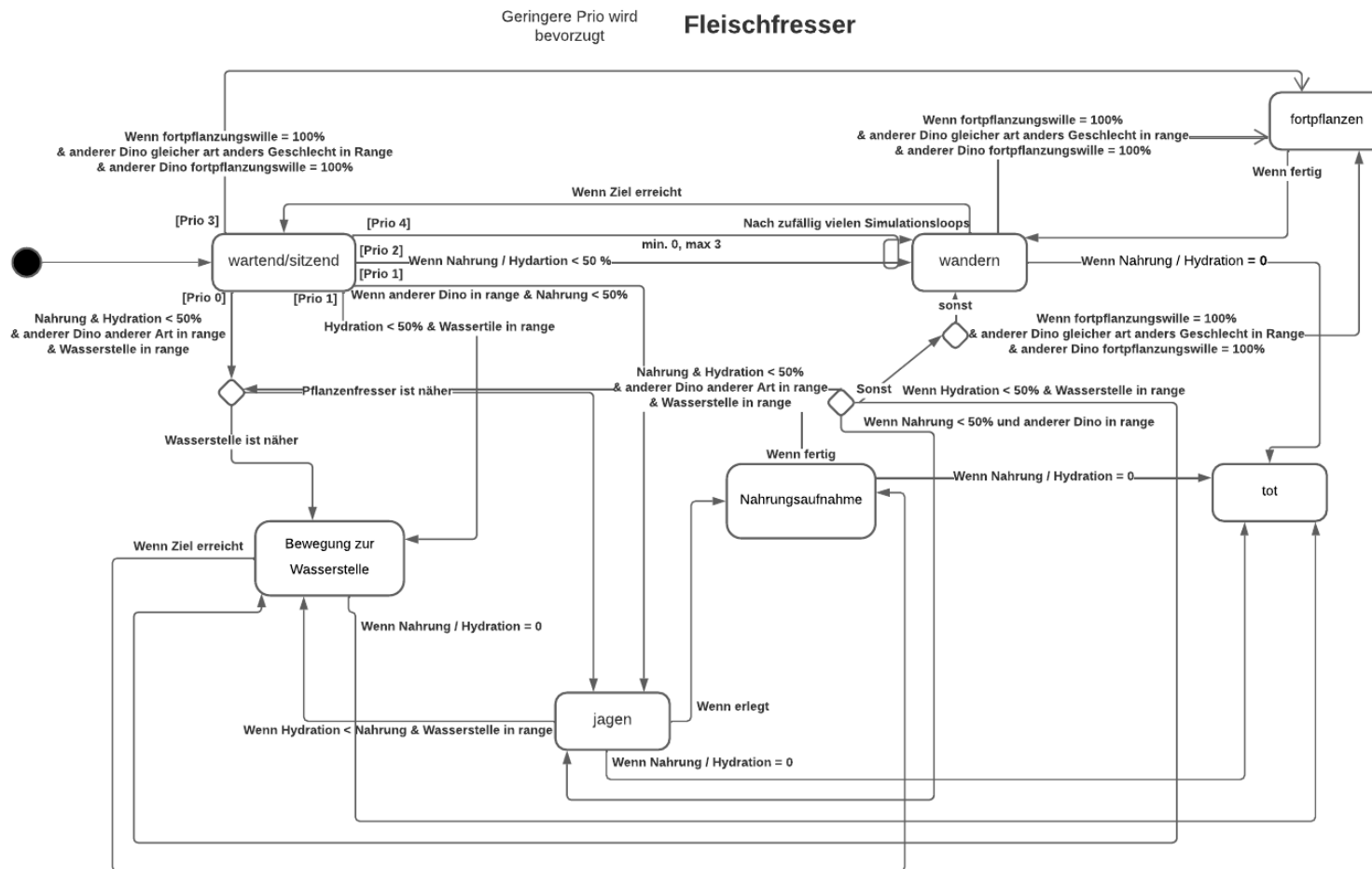



Abb. 4: Zustandsdiagramm Fleischfresser

Allesfresser

Der Allesfresser verhält sich ähnlich zu dem Fleischfresser, nur das er als zusätzliche Nahrungsquelle die Pflanzen besitzt. Der Allesfresser bevorzugt, wenn er die Wahl hat zwischen einem Pflanzenfresser hinterherzujagen und zu einer Pflanze zu gehen, welche sich beide in seiner Range (Line of Sight) befinden, immer zu der Pflanze zu gehen. Da dies noch einige Transitionen und Abzweige mehr bedeutet und damit das Zustandsdiagramm nur noch unübersichtlicher wird, wird hier darauf verzichtet ein separates Zustandsdiagramm für einen Allesfresser einzufügen, da dieses keinen wirklichen Mehrwert liefert.

7. Datenhaltung/Datenverarbeitung

Simulationsojekt-Konfiguration

Um die Lesbarkeit und Möglichkeit zur Bearbeitung (durch das Museumspersonal ohne eigene GUI) zu gewährleisten soll ein menschlich lesbares Datenformat gewählt werden. Daher fällt die Wahl auf das Datenaustauschformat „JavaScript Object Notation“ (kurz: JSON). 

Varianz bedeutet, dass Eigenschaften wie Hunger, Durst usw. ein Minimal- und Maximalwert angegeben haben.

Im nachfolgenden wurden außerdem die Begriffe Hunger zu Nahrung und Durst zu Hydration umbenannt.

- Hierfür wird nur der Import implementiert.
- Form der JSON-Datei = Schlüssel : Werte
 - Dinosaurierart :
 - Name : String
 - Bild : String
 - Nahrung : [Zahl als Minimalwert, Zahl als Maximalwert]
 - Hydration : [Zahl als Minimalwert, Zahl als Maximalwert]
 - Stärke : [Zahl als Minimalwert, Zahl als Maximalwert]
 - Geschwindigkeit : [Zahl als Minimalwert, Zahl als Maximalwert]
 - Fortpflanzungswilligkeit : [Zahl als Minimalwert, Zahl als Maximalwert]
 - Gewicht : Zahl
 - Länge : Zahl
 - Höhe : Zahl
 - KannSchwimmen : Boolean
 - KannKlettern : Boolean
 - Nahrungsart : Char (Pflanzenfresser, Fleischfresser, Allesfresser)
 - Sichtweite : [Zahl als Minimalwert, Zahl als Maximalwert]
 - Interaktionsweite : [Zahl als Minimalwert, Zahl als Maximalwert]
 - Pflanzenart
 - Name : String
 - Bild : String
 - Interaktionsweite : [Zahl als Minimalwert, Zahl als Maximalwert]

Szenariokonfiguration

- Hierfür werden Import sowie Export implementiert.
- Form der JSON-Datei = Schlüssel : Werte
- Anforderung an Mindestvorkommen der Parameter entnehme Kapitel 4.2 des Pflichtenhefts
 - Dinosaurier :
 - AnzahlDinosaurierartName : Zahl (als Anzahl der Dinosaurierart)
 - Pflanzen :
 - AnzahlPflanzenartName : Zahl (als Anzahl der Pflanzenart)
 - Pflanzenwachstum : Zahl (Wachstum für alle Pflanzen)
 - Landschaftsname : String

Landschaftskonfiguration

- Hierfür wird nur der Import implementiert.
- Form der JSON-Datei = Schlüssel : Werte
 - Landschaften :
 - Name : String
 - Aufbau: [[], [], ...] ?

Statistiken

Im Rahmen der Software werden zwei verschiedene Arten von Statistiken gepflegt und können zur Information des Nutzers angezeigt werden.

Einzelstatistiken beziehen sich auf einen einzelnen Dinosaurier, welcher beim Pausieren der Simulation angeklickt werden kann. Dabei werden schlicht die relevanten Attribute des entsprechenden Objektes (Eigenschaften des Dinosauriers) abgefragt und danach (beschriftet, seitlich) dargestellt.

Die Gesamtsimulationsstatistik wird am Ende jeder Simulation angezeigt. In ihr enthalten sind verschiedene Informationen wie die anfängliche Anzahl jeder Spezies, die übrig gebliebenen Exemplare zum Ende der Simulation, sowie die prozentuale Veränderung der Population. (siehe Abbildung 7)

Statistik Abbildung?

8. Simulationsloop

Automatisch

Im Folgenden wird mithilfe eines Sequenzdiagramms der Ablauf der Simulationsloop im automatischen Verfahren dargestellt.

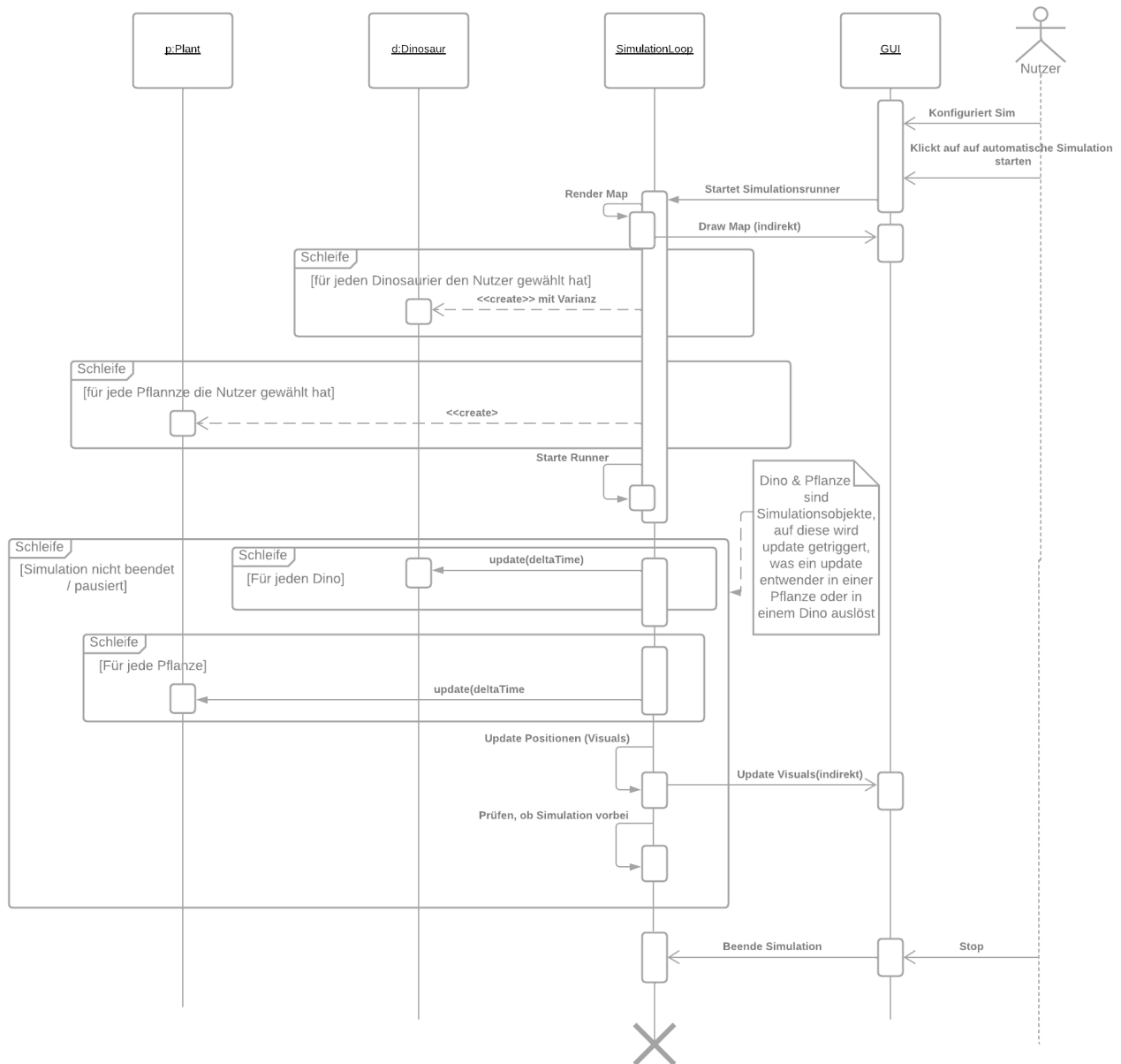


Abb. 5: Sequenzdiagramm-Simulationsloop-auto

Schrittverfahren

Im Folgenden wird mithilfe eines Sequenzdiagramms der Ablauf der Simulationsloop im Schrittverfahren dargestellt.

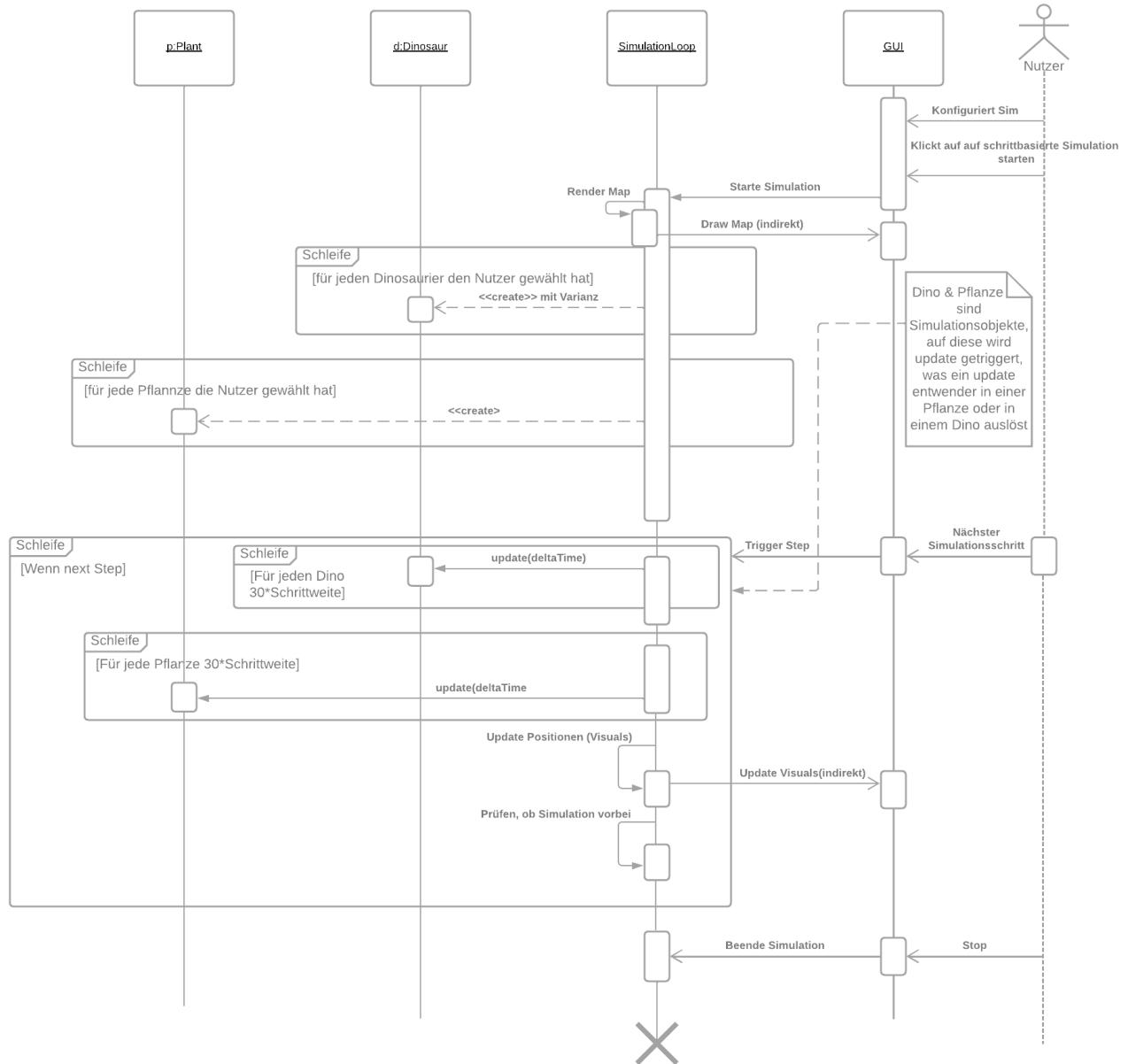


Abb. 6: Sequenzdiagramm-Simulationsloop-step

9. GUI

Die nachfolgenden Grafiken sind unverbindlich.

Das folgende Mockup stellt die spätere Simulationsansicht grob graphisch dar:



Abb. 7: Mockup - Simulationsoberfläche

Folgendes Mockup bildet die Darstellung der Konfigurationsansicht ab:

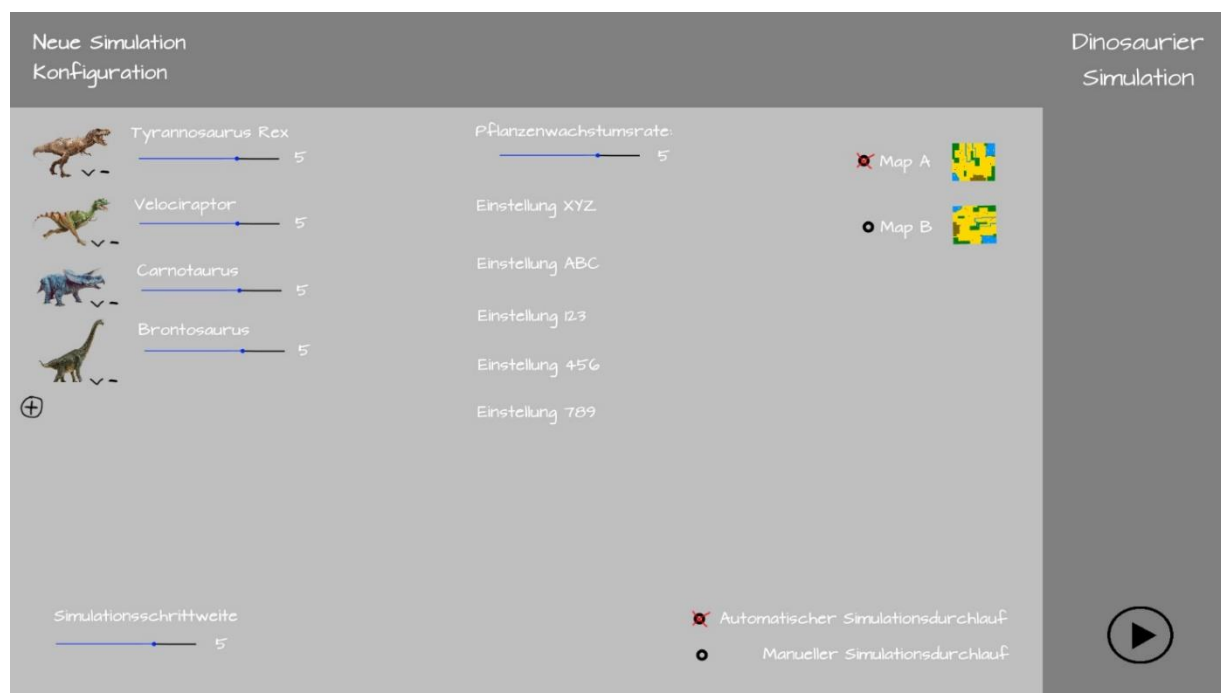


Abb. 8: Mockup - Konfiguration

Die Darstellung der Statistiken wird in folgender Grafik abgebildet:

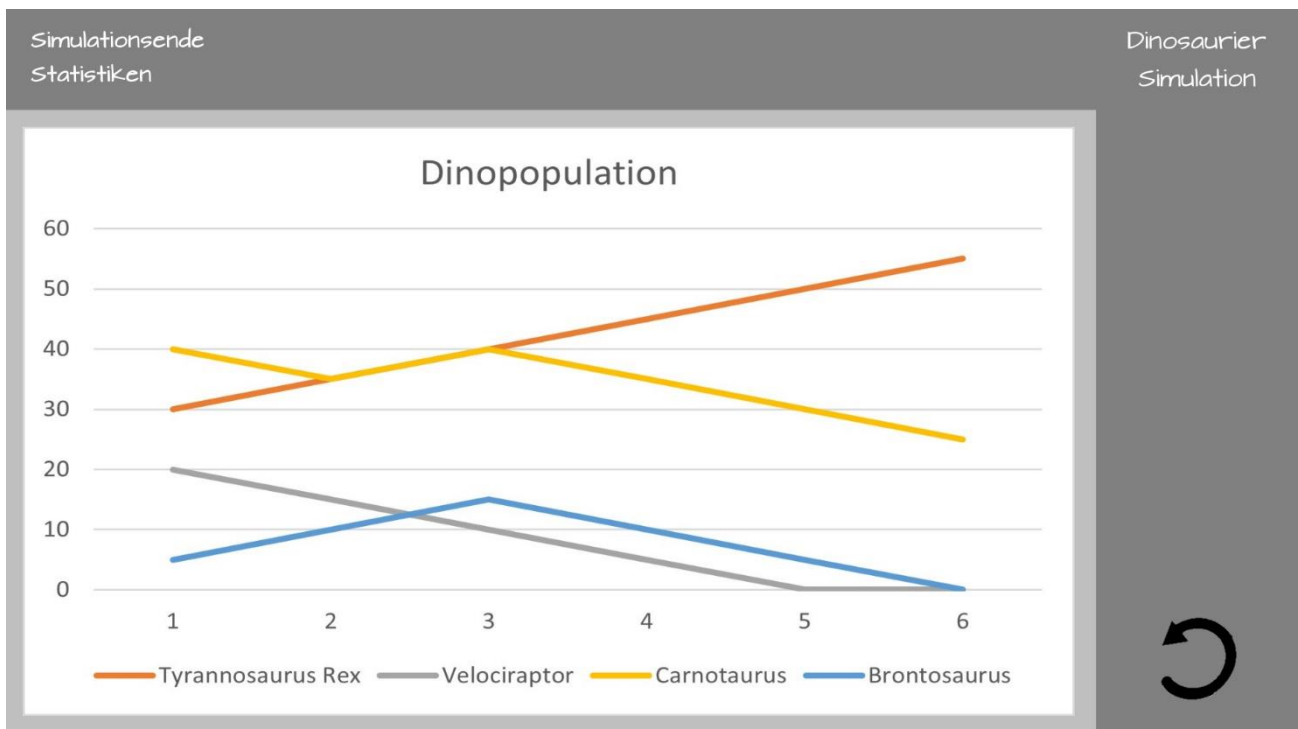


Abb. 9: Mockup - Statistiken