Universität Bremen

Interaktives Museumsexponat zur Darstellung simulierter Pflanzenevolution mit Hilfe des L-Systems

Bachelorarbeit

Simay Akin

Matrikel-Nr. 4508730

simay1@uni-bremen.de

Erstgutachter: Prof. Dr.-Ing. Udo Frese
Zweitgutachter: Dr.-Ing. Robert Porzel

26. Juni 2022

Selbstständigkeitserklärung

Hiermit versichere ich, dass ich die vorliegende Abschlussarbeit selbstständig angefertigt habe. Es wurden keine anderen als die angegebenen Quellen und Hilfsmittel benutzt. Sämtliche wissentlich verwendeten Textausschnitte, Zitate oder sonstige Inhalte anderer Verfasser, die wörtlich oder sinngemäß aus Veröffentlichungen entnommen wurden, sind als solche gekennzeichnet.

Bremen, den 26. Juni 2022

Simay Akin

Zusammenfassung

Das Ziel der vorliegenden Arbeit ist es, ein Museumsexponat zu programmieren, welches Menschen das Konzept der Evolution näherbringt. Dazu werden als Grundlagen das L-System von Aristid Lindenmayer und ein genetischer Algorithmus verwendet. Um die Usability des Exponats und die Qualität der implementierten Pflanzenevolution zu testen, wurde eine Evaluation durchgeführt. Die Evaluation zeigte,

Inhaltsverzeichnis

Abbildungsverzeichnis

1 Einleitung

Im Folgenden wird die Motivation und das Ziel der Arbeit dargestellt. Zudem werden die Schritte erläutert, die zum Ziel der Arbeit führen.

1.1 Motivation

Wenn die meisten Menschen den Begriff Evolution hören, denken sie an Darwins Theorie und an den damaligen Schulunterricht. Nach der Schule beschäftigen sich in der Regel nur Menschen, die sich für Evolution und Biologie interessieren mit diesen Themen. Dabei sind Evolution und Weiterentwicklung Themen, die selbst in Videospielen und anderen Computerprogrammen Verwendung finden. Seien es die Entwicklungen von kleinen Taschenmonstern in den weltweit beliebten *Pokémon* Spielen, oder die Kreuzung von Pflanzen, um neue Blütenfarben entstehen zu lassen, wie in *Animal Crossing: New Horizons*.

Doch auch für komplexere Prozesse erwies sich die Evolution als eine Methode, um diese zu erforschen, ohne dass menschliches Verständnis für jedes Detail des jeweiligen Prozesses gefordert wird. So wurden schon für Prozedurale Inhaltsgenerierung (Englisch: Procedural Content Generation) Prozesse verwendet, die von der Evolution inspiriert sind. Eine der bekanntesten Ansätze für diese Generierung ist das L-System, welches vom theoretischen Biologen Aristid Lindenmayer [3] entwickelt wurde. Dieses Modell spiegelt das Prinzip des Pflanzenwachstums in der Natur wider. Noch heute findet das L-System Anwendung in der Computergrafik. So wird es gerne genutzt, um natürliche Strukturen so wie Landschaften oder Höhlen zu generieren [4]. Auch die Modellierung und die Evolution von Pflanzen wurde bereits mehrfach untersucht, einer dieser Ansätze beinhaltet die Simulation von Pflanzlicher Evolution. Niklas [5] stellt die pflanzliche Evolution da, indem er verschieden Hypothesen über diese aufstellt. Um diese Hypothesen in messbare Größen umzuwandeln, benutzt er mathematische Verfahren. Projekte wie diese dienen der Untersuchung von Evolutionstheorien, indem diese computerbasiert dargestellt werden. Zur Darstellung und Vermittlung von Wissen über die pflanzliche Evolution mithilfe dieser verschiedenen Ansätze und Modelle gibt es nur wenig Konzepte, was die Hauptmotivation dieser Arbeit war.

1.2 Ziel der Arbeit

Ziel dieser Arbeit ist es ein interaktives Museumsexponat zu implementieren und mit diesem die pflanzliche Evolution simuliert darzustellen und den Menschen näherzubringen.

- Der erste Schritt ist es Pflanzen darzustellen. Dazu wird das L-System verwendet welches dabei hilft, die Gene zu interpretieren und visuell darzustellen. Um das

Exponat so naturgetreu wie möglich zu halten, werden Genotypen verwendet, welche das L-System dann zu Phänotypen umwandelt.

- Das zweite Ziel der Arbeit ist es den Evolutionsprozess naturgetreu umzusetzen, wozu ein genetischer Algorithmus verwendet wird. Dieser besteht aus drei Hauptteilen: Population erzeugen, Population evaluieren, neue Population erzeugen.
- Der dritte Schritt besteht darin, diese beiden Verfahren zu kombinieren und daraus ein Museumsexponat zu erstellen, welches die Evolution spielerisch umsetzt und eine interaktive Veränderung der Evolution ermöglicht.
- Wenn diese drei Schritte erfüllt sind, wird im Anschluss eine User Study durchgeführt, um zu testen, wie das Exponat in der Praxis funktioniert und ob die Evolution verständlich implementiert wurde.

2 Verwandte Arbeiten

Im folgendem Kapitel werden verwandte Arbeiten vorgestellt. In beiden Abschnitten wird jeweils ein Projekt veranschaulicht, welches ähnliche Ansätze zu dem Exponat beinhaltet, welches in dieser Arbeit vorgestellt wird.

2.1 AntFarm

AntFarm [1] ist ein fortlaufendes Projekt. Dieses Modell ermöglicht es eine simulierte Evolution von komplexen Verhaltensweisen in komplexen Umgebungen, sowie die Zusammenarbeit zwischen eng verwandten Individuen und der chemischen Kommunikation zu untersuchen.

AntFarm ist ein Computerprogramm, welches die Fortentwicklung von Nahrungssuchstrategien in Kolonien künstlicher Organismen, die Ameisen ähneln, simuliert. In ihrem Programm verwenden Robert J. Collins und David R. Jefferson [1] einen genetischen Algorithmus, um die Veränderung des Nahrungssuchverhaltens und die Strategie der Ameisen durch Selektionsdruck zu simulieren. Hierbei wird der Fortpflanzungserfolg durch die Menge von Nahrung die zum Nest transportiert und weiteren Faktoren, die im folgenden genauer beschrieben werden, beeinflusst.

Der genetische Algorithmus, der von Collins und Jefferson [1] angewendet wird, arbeitet auf der Ebene der Ameisenkolonien und nicht mit einzelnen Ameisen. Jede dieser Ameisenkolonien besteht aus identischen Ameisen; ihr Verhalten wird festgelegt durch künstliche neuronale Netze (KNN). Eine Ameise kann neben der Nahrungssuche und dem Nahrungstransport auch Pheromone verteilen, welche Ameisen in der Natur zur Kommunikation verwenden. Das Verhalten einer Ameise trägt zu der Fitness der Kolonie bei. Um das Verhalten anzuwenden, besitzt jede Ameise zwei KNN, welche miteinander verbunden sind. Das Suchnetzwerk wird aufgerufen, wenn die Ameise keine Nahrung trägt. Das Transportiernetzwerk wird aufgerufen, sobald die Ameise Nahrung gefunden und aufgehoben hat. Die Nahrungssuche besteht aus den Aufgaben Nahrung zu suchen und zum Nest zu transportieren. Transportiert die Ameise keine Nahrung, achtet sie auf Nahrung und Pheromone in ihrer Umgebung, auf die Nestposition und auf einen implementierten Kompasssensor. Trägt die Ameise Nahrung, achtet sie nur auf den Kompasssensor, um die richtige Richtung des Nestes feststellen zu können.

Die genetische Information der Startkolonien basieren auf zufälligen Chromosomen, welche durch Bits repräsentiert werden. Diese gesamte Population besteht aus 16,348 Kolonien, die jeweils aus 128 Ameisen bestehen. Insgesamt besteht die Population somit aus mehr als zwei Millionen Ameisen. Jede Kolonie befindet sich auf einem 16 x 16 Gitter. Die Fortpflanzung der Ameisenkolonien finden lokal auf diesen Gittern statt. Jede Kolonie hat die gleiche Anzahl an Nahrungselementen, auch die Verteilung ist für jede Generation gleich, damit keine Kolonie einen Vorteil hat. Jede Generation beginnt mit

der Erinnerung der Ameisen auf Null; jede Ameise lebt durch die gesamte Generation. Insgesamt lassen Collins und Jefferson [1] jede Generation für 100 Zeitschritte leben. In diesen Zeitschritten wird die Fitness der Kolonien durch die Ameisen beeinflusst. Diese Fitness wird im Selektionsprozess ausgewertet.

Jede Nahrungseinheit, die von einer Ameise zum Nest getragen wird, erhöht die Fitness um 1000 Punkte. Das Verteilen von Pheromonen und das Bewegen (Bewegen impliziert aufheben und ablegen) von Nahrung, minimiert die Fitness um jeweils 0.1 Punkte. Die Berücksichtigung dieser Punkte sorgt für den Selektionsdruck, durch welche die Nahrungssuchstrategien optimiert werden.

Die Eltern für die nächste Generation werden während der Nahrungssuche ausgewählt. Die Ameise mit der höchsten Fitnesspunktzahl wird selektiert. Die Selektionsphase des Algorithmus produziert ein Paar von Strings (Chromosome) für jedes Nachkommen, welche für die nächste Generation gelten. Es werden jeweils nur gleichgroße Stringelemente ausgetauscht. Das Modell der Rekombination, welches in *AntFarm* verwendet wird, beginnt mit der Ausrichtung der Stringpaare. Diese kreuzen sich an einer zufälligen Stelle, an dieser Stelle werden sie zertrennt und mit dem Stringelement des anderen Strings wieder zusammengefügt. Von den erzeugten Strings wird nur einer für die nächste Generation verwendet, alle weiteren werden verworfen. Diese Auswahl findet zufällig statt. Nach der Rekombination der Stringpaare, wird der verbliebene String durch eine Bit-Veränderung mutiert, welches zur Veränderung eines Bits führt (1 zu 0 oder 0 zu 1).

2.2 On Genetic Algorithms and Lindenmayer Systems

'On Genetic Algorithms and Lindenmayer Systems' ist ein Paper von Gabriela Ochoa [6] aus dem Jahr 1998. In diesem stellt sie ein System zur Darstellung simulierter Evolution von zwei dimensionalen Pflanzen vor.

In ihrem System benutzt Ochoa [6] wie Collins und Jefferson [1] einen genetischen Algorithmus und zusätzlich das Lindenmayer-System; entwickelt von Aristid Lindenmayer. Dieses System verfügt über zwei verschiedene Arten der künstlichen Evolution. Zum einen die interaktive Auswahl, die auf menschlicher Wahrnehmung basiert. Diese ermöglicht es den Benutzern, die simulierte Evolution auf bevorzugte Formen zu lenken. Zum Anderen wird eine automatisierte Evolution durch einen genetischen Algorithmus benutzt, welcher sich an Faktoren orientiert, die in der Natur einen Einfluss auf Pflanzenevolution haben.

Ochoa benutzt drei Hauptoperatoren: Das Crossover und zwei Arten der Mutation. Beim Crossover werden zwei Stringfragmente der Elternpflanzen ausgetauscht. Die Symbolmutation tauscht ein Symbol mit einem zufälligen Symbol aus. Die Blockmutation ersetzt einen ganzen Stringblock mit einem zufälligen Stringblock aus.

Die Fitnessfunktion orientiert sich an folgenden Faktoren, welche dazu führen sollen, dass die simulierte Evolution zu Strukturen führt, die natürlichen Pflanzen ähnelt:

- Phototropismus (a): Die Pflanzen, deren maximale y-Koordinate hoch ist, erhalten eine höhere Fitness, die mit einer niedrigen maximalen y-Koordinate, erhalten eine geringere Fitness. Dadurch soll der Wachstum der Strukturen in Richtung Licht forciert werden.
- Bilaterale Symmetrie (b): Die Werte der x-Koordinaten der Eckpunkte, jeweils links und rechts der vertikalen Achse werden addiert. Eine höhere Fitness erhalten die Pflanzen, die eine ausbalancierte Struktur haben.
- Lichtsammelvermögen (c): Die Blätter die nicht von anderen Blättern verdeckt sind, sind dem direkten Licht ausgesetzt. Die Pflanzen mit einer höheren Anzahl an dem Licht ausgesetzten Blättern, erhalten eine höhere Fitness.
- Strukturelle Stabilität (d): Die Verzweigungen an jedem Verzweigungspunkt werden gezählt. Eine höhere Anzahl an Verzweigungspunkten führt zu instabilität, daher werden Pflanzen mit einem hohen Anteil dieser Knoten niedrig bewertet.
- Anteil der Verzweigungspunkte (e): Die Verzweigungspunkte mit mehr als einem Ast werden berechnet, diese Zahl steht im Verhältnis zur Gesamtanzahl der Verzweigungen in einer Struktur. Pflanzen mit einer hohen Anzahl von Verzweigungen sind besser in der Lage, das Licht zu sammeln und Samen zu verbreiten.

Zusätzlich zu diesen Faktoren werden noch Gewichtungsparameter berücksichtigt, welche zur Berechnung der Stärke jedes Faktors benutzt werden. Die Berechnung ist wie folgt:

$$F(Phenotyp)=\frac{aw_a + bw_b + cw_c + dw_d + ew_e}{w_a + w_b + w_c + w_d + w_e} \tag{1}$$

Bei der Durchführung des genetischen Algorithmus wurden folgende Werte verwendet:

- Populationsgröße = 50
- Anzahl der Generationen = 100
- Generationslücke = 20%
- Chromosomenlänge = 7-20

Durch verschiedene Gewichtungen wurden verschiedene Pflanzenmodelle generiert:

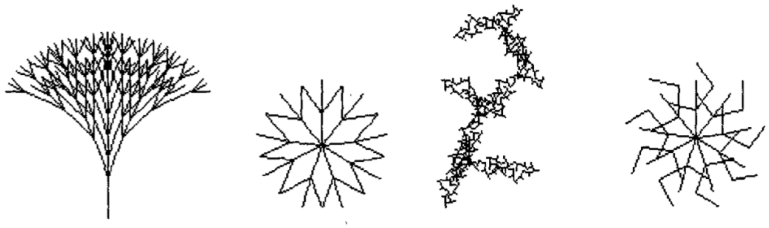


Abbildung 1: Durch eine balancierte Gewichtung erzeugte Pflanzen [6]

Bei den Pflanzen auf Abbildung 1 wurde eine Gewichtung von jeweils 50 für jeden Faktor der Fitnessfunktion verwendet. Bei einer Gewichtung von: a = 100, b = 90, c = 40, d = 20 und e = 30 wurden folgende Pflanzen generiert:

Abbildung 2: Durch eine unbalancierte Gewichtung erzeugte Pflanzen [6]

Ochoas [6] Simulationsergebnisse zeigen auf, dass das L-System ein geeigneter Ansatz ist um künstliche Evolution darzustellen.

3 Grundlagen

Bevor das Museumsexponat und dessen Funktion und Konzipierung vorgestellt werden, sind einige Grundkenntnisse erforderlich. In diesem Kapitel werden die zwei Grundlagen vorgestellt, welche in dem interaktiven Museumsexponat verwendet werden.

3.1 Genetischer Algorithmus

3.1.1 Grundstein des genetischen Algorithmus

Die Grundsteine für die genetischen Algorithmen die heute verwendet werden, wurden in den 1960ern von dem Informatiker John Holland [2] gelegt.

Das erste Resultat seiner Forschung zu einem geeigneten genetischen Algorithmus ist ein Klassifizierungssystem. Dieses System besteht aus einer Reihe von Regeln, von denen jede bestimme Aktionen ausführt, wenn ihre Regelbedingungen erfüllt sind. Diese Regeln und die Informationen die zur Erfüllung der Bedingungen erforderlich sind, werden durch Bitfolgen dargestellt. Für jede Bedingung die erfüllt wird, enthält ein String eine 1 an der entsprechenden Stelle und für jede nicht erfüllte Bedingung eine 0. Als Beispiel gibt Holland [2] eine Klassifizierungsregel an, die Hunde erkennt. Diese Klassifizierungsregel kann als ein String kodiert werden, der 1en für die Bits enthält, die Merkmale wie 'haarig', 'sabbernd', 'bissig' und 'böse' entsprechen und 0 für die Bits, die unpassende Merkmale enthalten wie 'metallisch', 'spricht Urdu' und 'besitzt Kreditkarten'.

Um Klassifizierungsregeln zu entwickeln, die ein bestimmtes Problem lösen, gibt Holland [2] den Ansatz mit einer Population zufälliger Strings aus 0en und 1en zu starten und diese nach der Qualität ihrer Ergebnisse zu bewerten. Das Maß an Eignung der Ergebnisse kann je nach Problem eine beliebige Anzahl an Kriterien sein. Die geeigneten Strings werden zur 'Paarung' ausgewählt, während die ungeeigneten Strings aussortiert werden. Im Laufe des Generationsprozesses werden die Strings mit den besseren Lösungsansätzen überwiegen und im Paarungsprozess werden diese immer wieder auf neue Weise kombiniert, wodurch noch geeignetere Strings entstehen. Die Paarung der Strings orientiert sich an der Kreuzung biologischer Chromosomen. Hierbei werden die zwei höchstrangigen Strings ausgewählt, diese werden aneinandergereiht und ein Punkt entlang der Strings wird gewählt, um diese miteinander zu tauschen und zwei Nachkommen zu erzeugen. Die neu erzeugten Strings ersetzen nicht die Elternstrings, sondern nur solche, welche bei der Bewertung schlecht abgeschnitten haben. Diese werden bei jeder neuen Generation eliminiert, damit die Populationsgröße gleich bleibt. Des weiteren stellt Holland (1992) [2] das Konzept der Mutation vor, diese tritt bei einem von 10.000 Bits auf und ändert diese von 0 auf 1 oder 1 auf 0. Die Funktion der Mutation führt zu keiner Verbesserung der Ergebnisse, sondern verhindert eine einheitliche Population, die nicht in der Lage ist sich weiterzuentwickeln.

3.1.2 Verwendung des genetischen Algorithmus in Computerprogrammen

Auch Daniel Shiffman (2012) [7] stellt einen Ansatz zur Benutzung von genetischen Algorithmen in Computerprogrammen vor. Er orientiert sich an der natürlichen Selektion nach Charles Darwin. Um die natürliche Selektion so naturgetreu wie möglich abbilden zu können, fokussiert er sich auf folgende drei Punkte:

- Vererbung: Im Programm muss ein Verfahren enthalten sein, das den Eltern ermöglicht, ihre Gene an die nächste Generation zu vererben.
- Variation: Die Merkmale, die in der Population vorkommen, müssen vielfältig sein. Andernfalls muss es ein Verfahren geben, mit dem Variation eingeführt werden kann. Ohne jegliche Vielfalt werden die Nachkommen immer gleich zu den Eltern und untereinander sein. So ist es nicht möglich, neue Kombinationen von Merkmalen zu erreichen und die Population kann sich nicht weiterentwickeln.
- Selektion: Das Programm muss einen Mechanismus enthalten, welcher es einigen Individuen der Population ermöglicht, als Eltern bestimmt zu werden und ihre Gene weiterzugeben und anderen wiederum nicht. Dies orientiert sich an *'Survival of the Fittest'* der Darwin'schen Evolutionstheorie. Die Individuen, die den gesetzten Anforderungen am wenigsten entsprechen, werden gefiltert.

Um sicherzugehen, dass der Genpool wirklich die Gene enthält, die benötigt werden, generiert Schiffman [7] die Population zufällig. So stellt er sicher, dass die Population genügend Variation enthält. Die digitale Information, die jedes Inidividuum der Population enthält, dient als Genotyp. Der Phänotyp ist der visuelle Ausdruck dieser Informationen. Um die anschließende Selektion durchzuführen, müssen diese Informationen bewertet werden, dafür wird eine Fitnessfunktion verwendet. Durch diese Funktion wird für jedes Individuum ein numerischer Wert erzeugt, welcher die Fitness jedes Mitglieds der Population beschreiben soll. Sobald die Fitness für jedes Individuum der Population feststeht, kann ein Paarungspool generiert werden. In diesen Paarungspool werden nur die Individuen mit geeignetem Fitnesscore aufgenommen. Wurden die geeigneten zwei Eltern ausgewählt, werden diese durch Crossover reproduziert. Ist dieser Schritt beendet, wird ebenfalls wie bei Holland [2] Mutation angewendet. Die Mutationsrate kann den Evolutionsprozess ankurbeln, ist diese zu hoch, kann sie diesen Prozess aber auch beeinträchtigen. Wurden alle diese Schritte durchgeführt, springt man mit der neuen Generation zu Schritt eins zurück und wiederholt den gesamten Prozess erneut.

3.2 Lindenmayer System

Im Jahr 1968 entwickelte der theoretische Biologe Aristid Lindenmayer [3] ein grammatikbasiertes System zur Modellierung von Pflanzen. Das sogenannte L-System basiert

auf dem Konzept des Umschreibens. Das Umschreiben ist eine Technik zur Definition komplexer Objekte durch sukzessives ersetzen von Teilstücken eines einfachen Objektes durch Umschreibregeln. Die L-Systeme haben die Besonderheit, dass sie anders als ähnliche Systeme, ihre Umschreibregeln parallel anwenden. Dies spiegelt die biologische Motivation von L-Systemen wieder, da dadurch die Zellteilung mehrzelliger Organismen dargestellt wird, bei denen viele Teilungen gleichzeitig stattfinden können.

3.2.1 DOL-System

Zu den einfachsten Klassen der L-Systeme gehört das sogenannte *DOL-System* (Deterministic Context Free System). Dieses System ist deterministisch und kontextfrei. Die Idee hinter diesem System erklären Lindenmayer und Prusinkiewicz [3] wie folgt:

Es sei V ein Alphabet, V^* die Menge aller möglichen Wörter aus V und V^+ die Menge aller nicht leeren Wörter aus V. Das OL-System ist ein geordnetes Triplett $G = \langle V, \omega, P \rangle$, wobei V das Alphabet des Systems ist, $\omega \in V^+$ ein nichtleeres Wort genannt Axiom und $P \subset V \times V^*$ eine endliche Menge an Produktionsregeln ist. Eine Produktionsregel $(a, x) \in P$ wird als $a \to x$ geschrieben. Der Buchstabe a und das Wort x werden Vorgänger und Nachfolger *(predecessor and successor)* genannt. Für jeden Buchstaben $a \in V$ gibt es mindestens ein Wort $x \in V^*$, so dass $a \to x$. Ist kein spezielles Wort x für einen bestimmten Vorgänger $a \in V$ gegeben, so wird $a \to a$ der Menge aller P hinzugefügt. Ein deterministisches OL-System, genannt DOL-System liegt dann vor, wenn es für jedes $a \in V$ ein und nur ein $x \in V^*$ exisitert, so dass $a \to x$.

Ein String besteht aus zwei Buchstaben (a und b), welche beliebig oft in diesem String vorkommen können. Jedem Buchstaben ist eine Umschreibregel zugeordnet. So bedeuetet die Regel $a \to ab$, dass der Buchstabe a durch die Zeichenfolge ab zu ersetzen ist. Die Regel $b \to a$ bedeutet, dass der Buchstabe b durch a ersetzt werden soll. Der Umschreibprozess beginnt mit einem Startwort, genannt Axiom, in diesem Beispiel b.

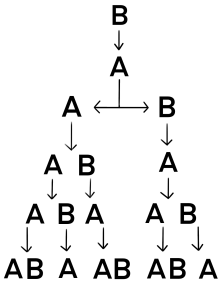


Abbildung 3: Beispiel der Umschreibregeln

Das Axiom wird im ersten Ableitungsschritt durch die Umschreibregel $b \to a$ durch den String a ersetzt. Im zweiten Schritt wird a durch ab ersetzt, der String ab besteht aus zwei Buchstaben, welche beide parallel im nächsten Ableitungsschritt ersetzt werden. Dadurch wird a durch ab ersetzt und b wird zu a, somit entsteht der String aba. Dieser Prozess wird beliebig oft durchgeführt, so dass am Ende im Beispiel der Abbildung 3 der String $abaababa$ entsteht. Diese Umschreibregeln sind ein erster Ansatz zur Darstellung von Baumstrukturen.

3.2.2 Turtle-Grafik

Um das DOL-System auf zwei und dreidimensionalen Strukturen anwenden zu können, benötigt man das Prinzip der Turtle-Grafik. Diese Grafik erwies sich als ideales System, um der Dynamik vom L-System eine geometrische Interpretation zu geben. Um verzweigte Strukturen darstellen zu können, muss das System erweitert werden. Die Grundlegende Idee präsentieren Lindenmayer und Prusinkiewicz [3] folgendermaßen:

Der Zustand der Turtle wird durch das 3-Tupel (x, y, α) definiert, wobei die kartesischen Koordinaten (x, y) die Position und der Winkel α die Richtung der Turtle angeben. Sind Schrittlängen d und Winkelerhöhung δ angegeben, können die Befehle wie folgt repräsentiert werden:

- F : Mache einen Schritt der Länge d. Der Zustand der Turtle ändert sich von (x, y, α) zu (x', y', α), wobei $x' = x + d\cos(\alpha)$ und $y' = y + d\sin(\alpha)$. Die Strecke wird zwischen Punkt (x, y) und (x', y') gezeichnet.
- f : Mache einen Schritt der Länge d ohne diese Strecke zu zeichnen.
- $+$: Drehe dich nach links um den Winkel δ. Der Zustand der Turtle ändert sich von (x, y, α) zu $(x, y, \alpha+\delta)$. Die positive Winkelorientierung ist gegen den Uhrzeigersinn.
- $-$: Drehe dich nach rechts um den Winkel δ. Der Zustand der Turtle ändert sich von (x, y, α) zu $(x, y, \alpha - \delta)$.

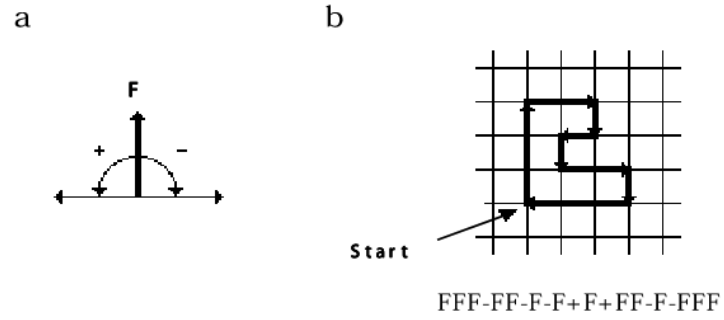


Abbildung 4: (a) Turtle-Interpretation der String Symbole F, +, −. (b) Interpretation eines Strings, die Winkelerhöhung δ entspricht 90° [3]

Die Turtle-Interpretation von einem String v, mit einem Anfangszustand von (x_0, y_0, α_0) und festen Paramtern d und δ ist die Zeichnung, die durch die im String angegeben Befehle entsteht (Abbildung 4). Diese Methode kann angewendet werden, um Strings, die durch ein L-System generiert werden, graphisch zu interpretieren.

3.2.3 Geklammerte L-Systeme

Mit den vorgestellten Befehlen in Abschnitt 3.2.1 und Abschnitt 3.2.2 sind durch die Turtle-Interpretation eines Strings nur Streckenzüge möglich, dabei ist das Ende einer Strecke mit dem Kopf der nächsten Strecke verbunden. Um auch Verzweigungen darstellen zu können, hat Lindenmayer [3] eine Notation mit Klammern für die Repräsentation von graphischen Bäumen vorgeschlagen. Diese Notation erweitert die Turtle-Interpretation von Strings.

Zur Abgrenzung einer Verzweigung wurden zwei neue Symbole eingeführt:

- [: Lege den aktuellen Zustand der Turtle auf dem Stack ab. Die auf dem Stack gespeicherten Informationen enthalten die Position und die Ausrichtung.
-] : Hole den obersten Zustand des Stacks und mache es zu dem aktuellen Zustand der Turtle. Die Position der Turtle verändert sich, ohne dass sie eine Linie zeichnet.

Diese Grundlagen dienten zum Aufbau meiner eigenen Konzeption, welche im folgenden Kapitel dargestellt werden.

4 Konzeption

In diesem Kapitel werden die Idee hinter dem interaktivem Exponat und das Konzept vorgestellt.

4.0.1 Idee

Die Grundidee, auf der das Exponat basiert, ist es Menschen das Konzept der Evolution näherzubringen. Wichtig war es einen Weg zu finden, dieses Konzept so umzusetzen, dass auch Laien, die wenig oder keine Berührungspunkte mit dem Thema Evolution haben, das Prinzip verstehen und Spaß an der Art der Vermittlung haben. Das Exponat soll also interaktiv und so gestaltet sein, dass es leicht verständlich ist.

Nach der Recherche über simulierte Evolution und ihre Umsetzung in Computerprogrammen, fiel meine Wahl auf das Lindenmayer System von Aristid Lindenmayer [3] und die Darstellung von pflanzlicher Evolution. Zusätzlich zu dem L-System habe ich einen genetischer Algorithmus umgesetzt, welchen ich nach dem Prinzip von Shiffman [7] programmiert habe. Wichtig war mir außerdem, die Phänotypen durch die Genotypen verändern zu können. Die Phänotypen sollen nicht durch das System festgelegt, sondern flexibel veränderbar sein.

4.0.2 Systementwurf

Das L-System ermöglicht die notwendige Unterscheidung zwischen Genotypen und Phänotypen und einen wohldefinierten Prozess um Phänotypen aus Genotypen zu erzeugen. Als Genotypen der Pflanzen habe ich folgende Attribute verwendet:

- Die verschiedenen Buchstaben und Zeichen (Umschreibregeln) des Alphabets des L-Systems.
- Die Farbe der Blüten der Pflanzen, wobei die Farbe des Stammes immer gleichbleibend ist.
- Die Anzahl der Iterationen, also die Anzahl der Wiederholungen der Umschreibregeln des L-Systems.
- Die verschiedenen Winkel der Abzweigungen.

Die erste Generation an Pflanzen startet mit zufälligen Werten dieser vier Attribute. Das L-System und die Funktionen des genetischen Algorithmus verarbeiten diese Informationen und stellen schließlich die Phänotypen dar. Die genaue Funktionsweise wird in Abschnitt 5 ausführlicher beleuchtet.

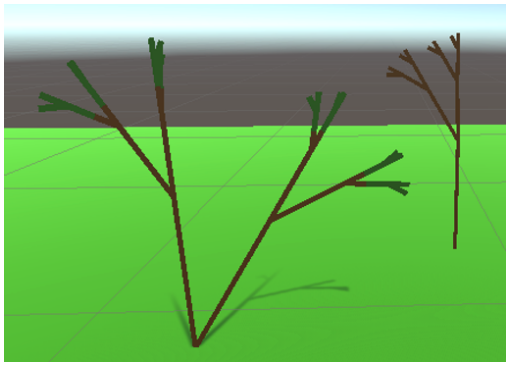


Abbildung 5: Erste Versuche von der Darstellung des Phänotyps.
Genotypen: [*FX][+FX] (links) und [F-[[X]+X]] (rechts)

Den Benutzer*innen soll es möglich sein, die Generationen durch ihre Präferenzen verändern zu können. Anders als bei dem Beispiel von Ochoa [6] in Abschnitt 2.2 wirken keine 'Natureinflüsse' auf die Pflanzen ein, sondern die Benutzer*innen können die Pflanzen gießen, indem sie diese anklicken. Dadurch erhöht sich die Fitness der Pflanzen, welche ihnen ein Vorteil im Selektionsprozess bringt. Die Benutzer*innen entscheiden, wann der Selektionsprozess beginnt und eine neue Generation gebildetet wird. Dabei habe ich einen Cooldown von 20 Sekunden zwischen jeder neuen Generation gesetzt, um ein direktes Erzeugen einer neuen Generation zu verhindern.

Für den Selektionsprozess habe ich mich für die sexuelle Fortpflanzung entschieden. Es werden die zwei fittesten Pflanzen ausgewählt, die ihre Gene kreuzen. Bei diesem Crossover wird ein Part der Gene eines Elternteils mit einem Part der Gene des anderen Elternteils ersetzt. Dabei wird bereits zu Beginn entschieden, welcher DNA-String weitergegeben wird. Aus diesem String entsteht dann die Kindspflanze der nächsten Generation. Sobald eine neue Generation erzeugt wird, wird die schwächste Pflanze aussortiert und verschwindet aus dem Programm. Die Populationsgröße bleibt dabei gleich.

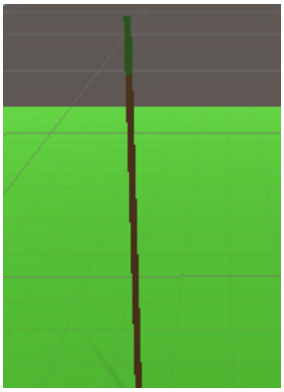


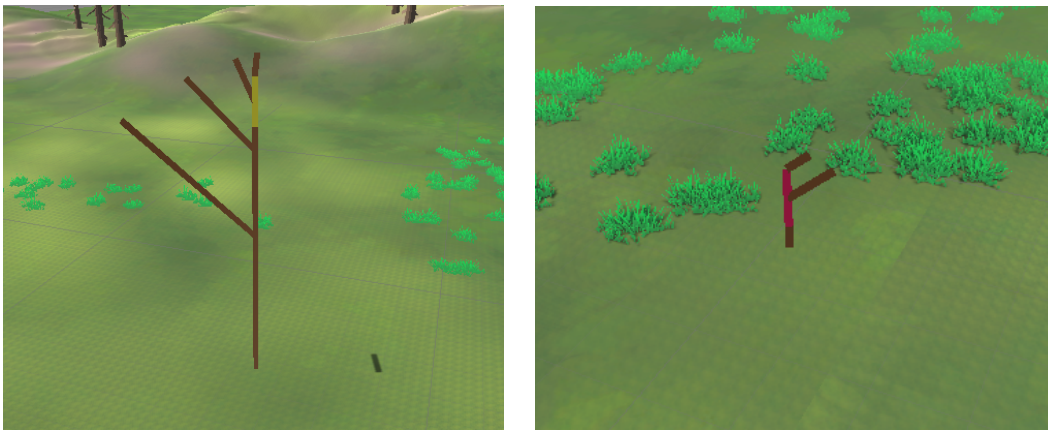
Abbildung 6: Kindspflanze der Pflanzen aus Abbildung 5. Genotyp: [*FX]+

Abbildung 6 zeigt ein Beispiel einer Kindspflanze einer neuen Generation, die aus dem Selektionsprozess entstanden ist. Dabei wurde ein Teil der Gene des ersten Elternteils (Abbildung 5) durch einen Teil des zweiten Elternteils ersetzt. Der DNA-String änderte sich wie folgt:

$$[*FX][+FX] \rightarrow [*FX]+$$

Das hintere Stück '[+FX]' des DNA-Strings der ersten Elternpflanze wurde mit dem hinteren Stück '+' des zweiten Elternteils ersetzt.

Vor der Generierung einer Kindspflanze gibt es noch den letzten Schritt der Mutation. Diese wird nicht bei jeder neuen Generation angewendet, sie hat nur eine Wahrscheinlichkeit von 10%. Durch die Mutation wird sichergestellt, dass die weitergegebenen Gene nicht gleich bleiben, sondern sich auch verändern und Variation bekommen. Die Mutation kann Gene entfernen, aber auch neue hinzufügen. Die verschiedenen Fälle der Mutation haben eine gleich hohe Wahrscheinlichkeit.



(a) Erstes Elternteil mit Genotyp: [*FX[+F]] und vier Iterationen

(b) Zweites Elternteil mit Genotyp: [*FX[+F]] und zwei Iterationen

Abbildung 7: Pflanzeneltern für das Beispiel der Mutation

In Abbildung 7 sieht man zwei Elternpflanzen, welche für die nächste Generation an Pflanzen ausgewählt wurden. Sie haben dieselben Umschreibregeln, werden aber durch das L-System trotzdem unterschiedlich dargestellt, da die Anzahl ihrer Iterationen unterschiedlich sind. Diese kreuzen ihre Gene mit folgendem Ergebnis:

$$[*FX[+F]] \rightarrow [*FX[[*FX]+F]]$$

Bevor dieser DNA-String für das Kind weitergegeben wird, mutiert er in diesem Beispiel folgendermaßen:

$$[*FX[[*FX]+F]] \rightarrow [*FX[[*F]+F]]$$

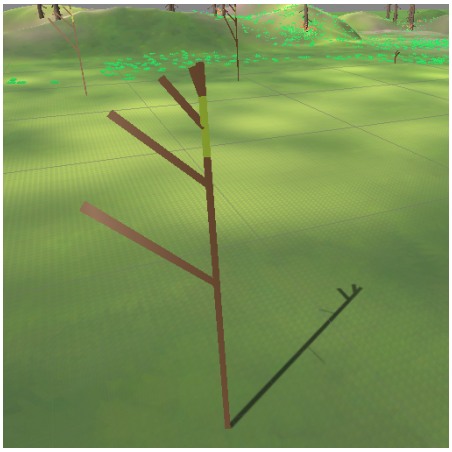


Abbildung 8: Pflanzenkind entstanden durch die Eltern aus Abbildung 7 und der oben beschriebenen Mutation

Aus dieser Mutation entsteht die Kindspflanze in Abbildung 8. In diesem Beispiel entfernt die Mutationsfunktion den Buchstaben 'X', statt einen neuen hinzuzufügen. Die Kindspflanze weist minimale Veränderungen in ihrer Spitze auf. Sichtbare Unterschiede werden teilweise erst nach einigen Generationen sichtbar.

Zusätzlich zu der Möglichkeit die Fitness der Pflanzen zu bestimmen und neue Generationen zu generieren, haben Benutzer*innen die Möglichkeit den gesamten Prozess zu verfolgen. Da das System als Museumsexponat konzipiert wurde, soll der Generationsprozess über mehrere Stunden laufen. Um diesen Prozess und seine verschiedenen Generationen betrachten zu können, habe ich in dem System eine Screenshot- und eine Galeriefunktion implementiert. Mit diesen Funktionen können die Benutzer*innen selber Fotos von ihren generierten Pflanzen schießen und diese und die Fotos der anderen Benutzer*innen in der Galerie betrachten. Jedes Bild, welches in der Galerie abgespeichert wird, zeigt die Uhrzeit zum Zeitpunkt der Aufnahme an, um den Zeitstrahl nachvollziehen zu können.

5 Implementierung

Das System wurde mit der Unity Version 2020.3.23f1 und der Programmiersprache C# implementiert.

5.0.1 Pflanzen

Um die verschiedenen Pflanzengene akkurat präsentieren zu können und jeder Pflanze ihre eigenen Gene zu geben, habe ich eine gesonderte Pflanzenklasse erstellt. Diese beinhaltet folgende Attribute:

- (a) Der DNA-String (Umschreibregeln)
- (b) Die vier verschiedenen Winkelwerte als *Float*
- (c) Der Wert für den Farbton der Blüten als *Float*
- (d) Zwei Listen vom Typ *String* und *Float* zum Speichern der einzelnen Buchstaben der Umschreibregeln und ihre zugehörigen Werte
- (e) Der Fitnesscore als *Integer*

Die Werte (a) - (c) werden zur Initialisierung der Ursprungspopulation zufällig festgelegt. Dabei liegt der Wert der Iterationen zwischen 2-5, die Werte der Winkel zwischen 0-60 und der Wert des Farbtons zwischen 0.0 - 1.0. Diese Werte werden den verschiedenen Buchstaben des DNA-Strings zugeordnet, indem alle Werte in den zwei vorhandenen Listen gespeichert werden. Diese zwei Listen haben also immer dieselbe Größe, um die Werte richtig zuzuordnen. Der Iterationswert wird an der Positions des Axiom 'X' gespeichert, der Farbwert an der Positions des Buchstabens 'F' und die Winkelwerte an den Positionen der Zeichen, die für die Winkel stehen. Somit bleiben die Werte, die aus dem Ursprungsstring durch Selbsteinsetzung entstehen, erhalten und es bleibt eine gewisse Selbstähnlichkeit erhalten. Sobald Teilstrings beim Crossover ersetzt werden, werden auch ihre zugehörigen Parameter ersetzt. Somit bestehen in der Kindspflanze nicht nur eine Mischung der *Strings*, sondern auch der numerischen Werte. Zusätzlich wird bei jedem Vererbungsprozess etwas Rauschen mitvererbt, so dass die Werte sich über die Zeit auch verändern. Bei diesem Rauschen werden die numerischen Werte erhöht oder minimiert, je nachdem welcher DNA-String für die Kindspflanze ausgewählt wird.

Das Weitervererben mit den Listen funktioniert folgendermaßen:

- DNA-String der Elternpflanze 1: [FX-]
- Werteliste der Elternpflanze 1: (0, 0.5, 2, 14.5, 0)

- DNA-String der Elternpflanze 2: [FX*]
- Werteliste der Elternpflanze 2: (0, 0.86, 4, 22.8, 0)

Ändert sich der DNA-String der ersten Elternpflanze durch den Vererbungsprozess folgendermaßen:

$$[\text{FX-}] \rightarrow [\text{FX}[\text{FX*}]]$$

Wird ebenfalls die Werteliste neu generiert und an die Kindspflanze weitergegeben:

$$(0,\ 0.5,\ 2,\ 14.5,\ 0) \rightarrow (0,\ 0.5,\ 2,\ 0,\ 0.86,\ 4,\ 22.8,\ 0)$$

Vor der Weitergabe dieser Liste werden die Winkelwerte noch leicht verändert, es wird entweder ein Rauschen zwischen 0.01 - 0.05 auf die Winkelwerte addiert oder subtrahiert. Wenn in einer Pflanze Zeichen doppelt auftauchen, die aber nur einmal verwendet werden können wie die Iteration, wird immer der letzte Wert übernommen, in diesem Beispiel wäre die Anzahl der Iterationen also 4.

Um den Benutzer*innen ein visuelles Feedback zu geben, wenn sie mit den Pflanzen inteagieren und diese gießen, habe ich für jede Pflanze ein *ParticleSystem* initialisiert. Das *ParticleSystem* ist eine eingebaute Funktion von Unity, die es ermöglicht verschiedene Arten von Partikeln einzubauen. Sobald ein*e Benutzer*in eine Pflanze erfolgreich angeklickt hat, spielt das *ParticleSystem* eine Animation der Partikel für einige Sekunden ab.

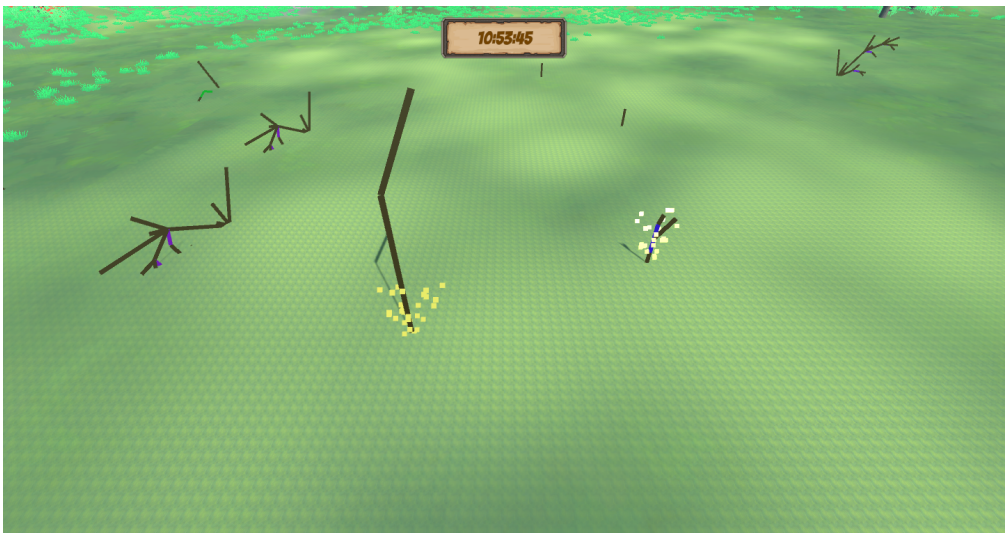


Abbildung 9: Visuelles Feedback für erfolgreiches Klicken, aufgenommen von einem*r Benutzer*in während der User Study

5.0.2 L-System

Um das L-System zu implementierern wird ein Dictionary und eine Funktion benötigt, die die Umschreibregeln ausführt. Durch das Dictionary lege ich fest, welche Buchstaben und Zeichen durch welche Umschreibregeln ersetzt werden sollen. Das Axiom 'X' wird immer durch den gesamten DNA-String ersetzt, beispielsweise: 'X → [FX+]'. Der Buchstabe 'F' wird jedes Mal durch 'FF' ersetzt, somit wird '[FX+]' → '[FF[FX+]+]'. Je nach Iteration wird dieser Schritt in der L-System Funktion wiederholt, bevor der String dann graphisch interpretiert wird.

In der L-System Funktion habe ich folgende Aktionen für folgende Buchstaben und Zeichen festgelegt:

- F: Initialisiert Stiele und ihre Blüten an der jeweiligen Position
- X: Für das Axiom wird keine Aktion durchgeführt, da dieses nur für die Umschreibregeln benötigt wird
- +: Rotiert den Ast um den angegeben Pluswinkelwert nach hinten
- -: Rotiert den Ast um den angegeben Minuswinkelwert nach vorne
- /: Rotiert den Ast um den angegeben Slashwinkelwert nach unten
- *: Rotiert den Ast um den angegeben Sternchenwinkelwert nach oben
- [: Legt die aktuelle Positions auf dem Positionsstack ab
-]: Holt den obersten Zustand des Stacks und macht sie zu dem aktuellen Zustand

5.0.3 Genetischer Algorithmus

Der genetische Algorithmus meines Programms besteht aus drei Hauptteilen: Initialisierung der Population, dem Selektionsprozess und der Mutation.

Die Initialisierung der Population beginne ich mit dem Aufrufen der L-System Funktion und der Zuordnung der numerischen Werte. Diese Funktion wird sooft aufgerufen, bis die richtige Anzahl an Pflanzen auf dem Spielfeld stehen. Diese können bis zu den Spielfeldgrenzen zufällig wachsen. Die Population wird durch die Fitness Funktion evaluiert. In der Funktion werden die Fitnessscores aller Pflanzen überprüft und die Liste der Population wird nach diesen Werten neu sortiert. Die schlechtesten Pflanzen werden nach jeder neuen Generation aussortiert, so dass die Population gleichbleibend groß ist.

Nach der Evaluation der Population werden die fittesten zwei Pflanzen als Elternpflanzen ausgewählt. Aus beiden DNA-Strings wird eine zufällige Position ausgewählt und das

Zeichen an dieser Stelle wird dann durch ein Zeichen des Strings der anderen Elternpflanze ersetzt. Da jede Generation nur eine Kindspflanze entsteht, wird ein DNA-String verworfen. Diese Auswahl findet zufällig statt. Nachdem der DNA-String für die Kindspflanze gewählt wurde, läuft diese in 10% der Fällen durch die Mutationsfunktion. Hier können die Zeichen F, X, +, -, * und / hinzugefügt oder entfernt werden. Die Klammern [und] werden dabei nicht beachtet. Jeder dieser Fälle hat eine gleich große Wahrscheinlichkeit. Nach dem Genaustausch und ebenfalls nach der Mutation werden die zwei Listen aktualisiert, um die neuen und verbliebenen Werte richtig zu positionieren. Zeitgleich zum Durchlauf der Umschreibregeln wird durch die Wertliste iteriert um die Variablen der Werte zu aktualisieren, so dass bei jedem Zeichen der richtige Wert benutzt wird.

5.0.4 Probleme während der Implementierung

Die erste Fortpflanzungsfunktion die ich implementiert habe wählte einen zufälligen Punkt in dem DNA-String aus, um diesen zu splitten. Anschließend wurden beiden Teilstücke der DNA-Strings zusammengefügt. Dadurch gab es in einigen DNA-Strings eine unbalancierte Anzahl an Klammern, was zu Problemen der Darstellung der Pflanzen führte. Aufgrund dessen konnte der *LineRenderer* den ich zum Abbilden der Pflanzen benutzt habe, die Positionen nicht mehr richtig nachvollziehen. Um diesem Problem vorzubeugen, wurde die Fortpflanzungsfunktion überarbeitet.

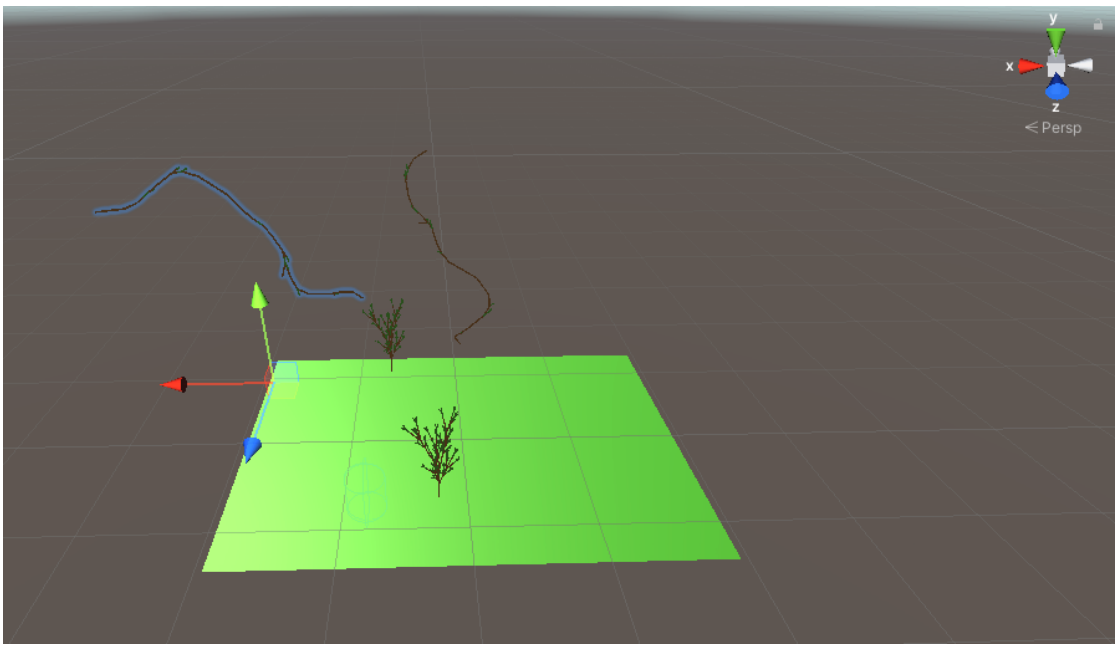


Abbildung 10: Pflanze generiert durch unbalancierten Klammerstring

Die überarbeitete Funktion funktioniert folgendermaßen:

```
GetBalancedSubstring:

    1. Suche einen zufälligen Index aus der Länge des unbalancierten
       Strings
    2. Wähle das Zeichen an diesem Index aus
    3. Wenn dieses Zeichen weder '[' noch ']' entspricht,
       überschreibe den Rückgabestring mit diesem Zeichen,
       ansonsten springe zu Punkt 5
    4. Speichere den Index als firstIndexOfSubString, springe zu
       Punkt 7
    5. Wenn das Zeichen '[' entspricht, iteriere durch den String:
            - Erhöhe den bracketCounter für jede '['
            - Verringere den bracketCounter für jede ']'
            - Ist der bracketCounter == 0 wurde die passende Klammer
              gefunden, die die Startklammer schließt
            - Überschreibe den Rückgabestring mit dem Substring, der
              von der Position der offenen Klammer bis zur
              geschlossenen Klammer reicht
            - Speichere den Index als firstIndexOfSubString
    6. Wenn das Zeichen ']' entspricht, iteriere rückwärts durch
       den String:
            - Verringere den bracketCounter für jede '['
            - Erhöhe den Counter für jede ']'
            - Ist der bracketCounter == 0 wurde die passende Klammer
              gefunden, die die Startklammer schließt
            - Überschreibe den Rückgabestring mit dem Substring, der
              von der Position der offenen Klammer bis zur
              geschlossenen Klammer reicht
            - Speichere den Index als firstIndexOfSubString
    7. Speichere den Rückgabestring und den firstIndexOfSubString in
       einem Tupel
    8. Gebe das Tupel zurück
```

Das zweite Problem trat durch das Einführen der zwei Listen ein. Da die Umschreibregeln durch das Dictionary festgelegt und mit einem *StringBuilder* diese Regeln angewandt wurden, konnten die Listen nicht angepasst werden und behielten ihre Ursprungsgröße. Somit war es nicht möglich, den DNA-String und die Werteliste parallel zu iterieren. Um dieses Problem zu umgehen, habe ich eine neue Funktion eingeführt. Diese führt die Umschreibregeln und das korrekte Erweitern der Listen durch, damit an den richtigen Positionen die richtigen Werte verwendet werden.

5.0.5 User Interface

Um das Programm so Benutzer*innenfreundlich wie möglich zu halten, habe ich das User Interface übersichtlich und simpel gehalten. Die Buttons haben keine Beschriftungen, um die Interaktion mit dem Programm zu fördern und den Benutzer*innen die Annäherung an die Funktionen selbstständig zu ermöglichen. Alle Buttons und andere Bilder sind kostenlose Assets aus dem Unity Asset Store.

Folgende Funktionen habe ich für das User Interface implementiert:

Abbildung 11: User Interface

- (a): Textfeld um die aktuelle Uhrzeit anzuzeigen
- (b): Button der zum Funktionsmenü führt
- (c): Button der zum Infomenü führt, wo Benutzer*innen Informationen über das Exponat erfahren
- (d): Button der zur Fotogallerie führt, in welcher die aufgenommenen Bilder der Benutzer*innen ausgestellt werden
- (e): Button mit dem neue Generationen generiert werden können, mit integriertem Generationszähler und einem Cooldown von 20 Sekunden, um ein schnelles wiederholtes Drücken des Buttons zu verhindern

Abbildung 12: Steuerungsmenü

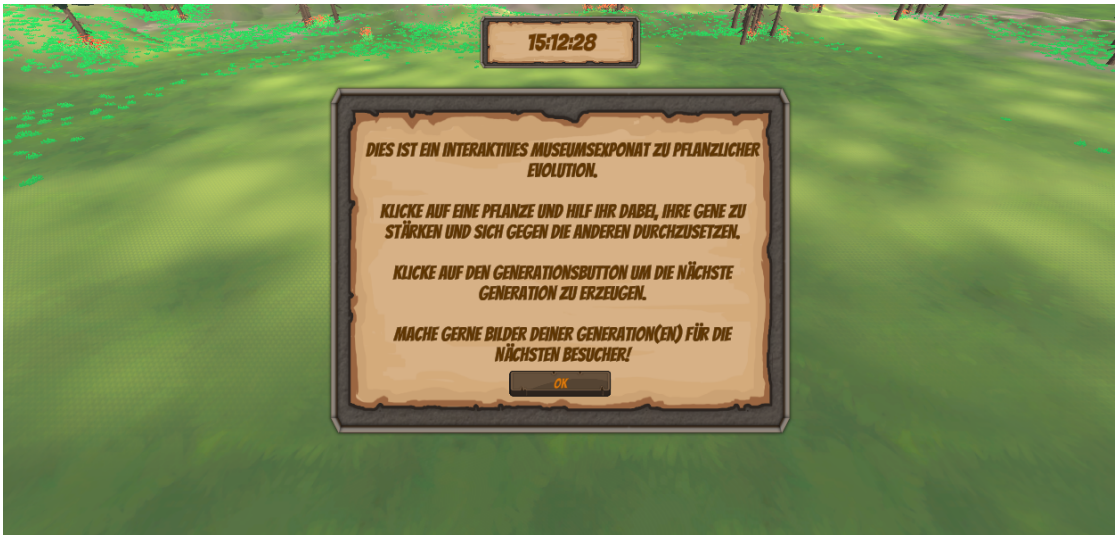


Abbildung 13: Infomenü

Abbildung 14: Fotogalerie

5.0.6 Spielumgebung und Sounds

Die Spielumgebung baute ich mit dem kostenlosen Terrain Sample Asset Pack aus dem Unity Asset Store. Das Hauptfeld, auf welchem die Pflanzen erzeugt werden (Abbildung 11), habe ich bewusst leer gelassen, um die Benutzer*innen nicht zu verwirren. Stattdessen entschied ich mich dafür im Hintergrund Bäume und kleine Hügel einzufügen. Um die natürliche Umgebung zu unterstützen, implementierte ich einen Natursound, welcher im Hintergrund des Programms spielt. Dieser stammt ebenfalls aus dem Unity Asset Store. Zusätzlich zu dem visuellen Feedback fügte ich Klicksounds ein, die abgespielt werden, wenn die Benutzer*innen etwas in dem Programm erfolgreich anklicken oder auswählen.

6 Evaluation

Zur Bewertung des Programmes, wurde im Anschluss an die praktische Anwendung eine User Study durchgeführt. Hierraus soll hervorgehen, ob die eingebauten Features wie gewünscht umgesetzt und genutzt wurden.

6.0.1 Pilotstudie

Vor der eigentlichen Studie entschied ich mich dazu eine Pilotstudie abzuhalten, um mögliche Bugs des Programms zu finden und ggf. zu beheben. In dieser Pilotstudie nahmen insgesamt fünf Teilnehmer*innen teil. Nach der 7. Generation konnte ein Bug entdeckt werden. Einige Pflanzen wuchsen umgekehrt in die negative Richtung der Y-Achse, statt in die Positive, was darin resultierte, dass einige Pflanzen unter dem Spielfeld lagen und für die Teilnehmer*innen nicht mehr sichtbar waren.

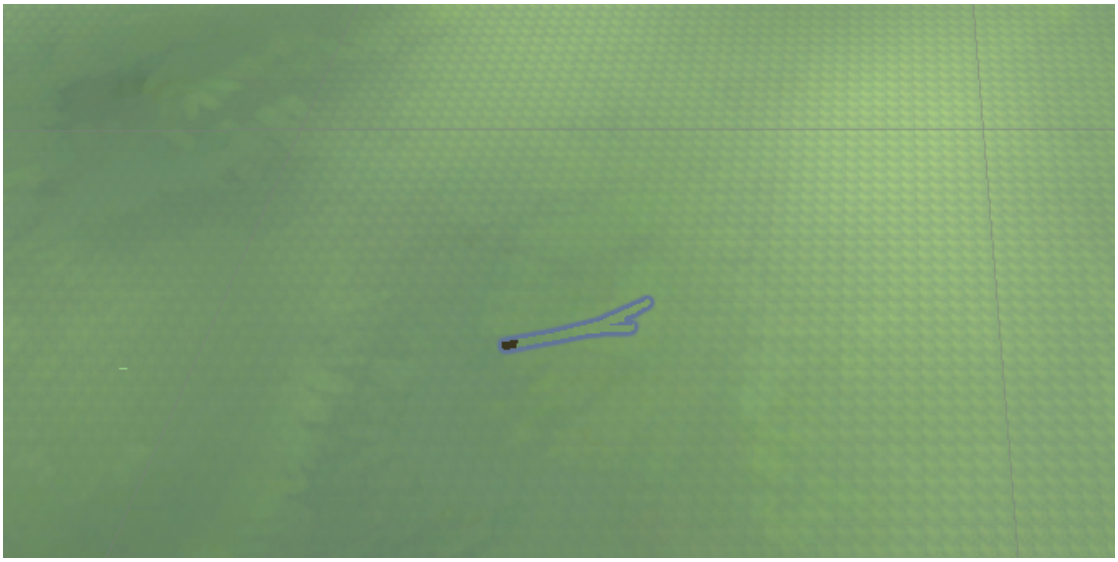


Abbildung 15: Pflanze die in den Boden wächst

Grund dafür war der *LineRenderer* aus Unity, welcher mit zwei 3D Vektoren arbeitet. Der Vektor 0 setzt die Startpositionen für jede Achse und der Vektor 1 die Endpositionen, diese werden dann genutzt um eine Linie zu zeichnen. Das Problem konnte ich beheben, indem der Vektor 0 vor jeder Generierung einer Pflanze nochmals auf 0 gesetzt wird. Aufgrund des Bugs wurden die Teilnehmer*innen der Pilotstudie nicht darum gebeten, den Fragebogen auszufüllen.

6.0.2 Studiendesign

Um den Aufbau eines Museumsexponats zu simulieren, habe ich das Programm im Eingangsbereich des Mehrzweckhochhauses der Universität Bremen ausgestellt. So konnten Menschen beim Vorbeigehen selber entscheiden, ob sie das Exponat ausprobieren wollen.

Abbildung 16: Aufbau der Studie

Um einschätzen zu können, wie gut das Programm das Prinzip der Evolution darstellt, wurden den Teilnehmer*innen im Anschluss 8 Fragen gestellt, welche ihre Nutzer*innenerfahrung festhalten. Zusätzlich wurde während der Programmnutzung die Zeit und die Anzahl der Interaktion (i.F.v. Mausklicks) gemessen. Ferner werden noch die Bilder der Teilnehmer*innen ausgewertet um zu analysieren, wie gut das Programm die Evolution umsetzt. Vor der Studie habe ich den Teilnehmer*innen keine Anweisungen gegeben, da alle Informationen in dem Programm zu finden sind. Insgesamt lief die Studie über 6 Stunden verteilt.

6.0.3 Studienergebnisse

Im folgenden werden die Ergebnisse der Umfrage vorgestellt. Insgesamt spielten 20 Teilnehmer*innen das Spiel und beantworteten den zugehörigen Fragebogen. Zwar erlaubt diese Anzahl an Teilnehmer*innen keine tiefgehende Analyse, sie gibt aber einen guten Eindruck davon, wie das Exponat aufgenommen wurde und welche Aspekte noch verbessert werden könnten.

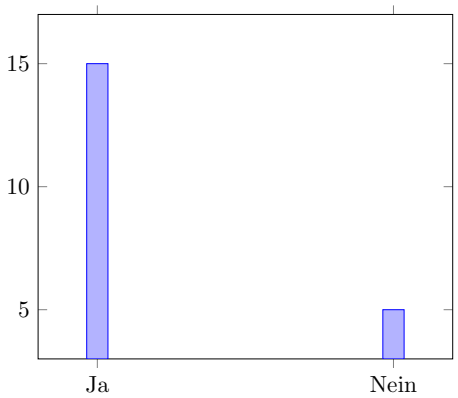


Abbildung 17: „Ich habe mich schon mal mit (pflanzlicher) Evolution befasst (Schule, YouTube, Fernsehen etc.)“

Wie Abbildung 17 zeigt, gab eine große Anzahl der Teilnehmer*innen an, sich bereits mit Evolution außeinandergesetzt zu haben. Fünf Teilnehmer*innen gaben an, sich noch nie mit Evolution befasst zu haben. Daraus lässt sich schließen, dass die Mehrzahl der Anwender*innen bereits vertraut mit der Materie waren.

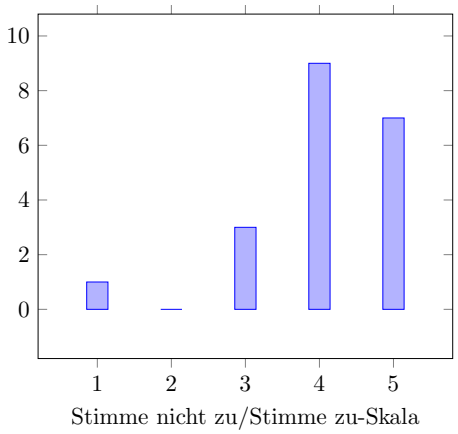


Abbildung 18: „Ich bin mit dem Museumsexponat gut zurechtgekommen“

In Abbildung 18 ist zu sehen, dass ein Großteil der Teilnehmer*innen mit dem Exponat gut zurechtgekommen sind. Lediglich eine Person stimmte der Aussage nicht zu.

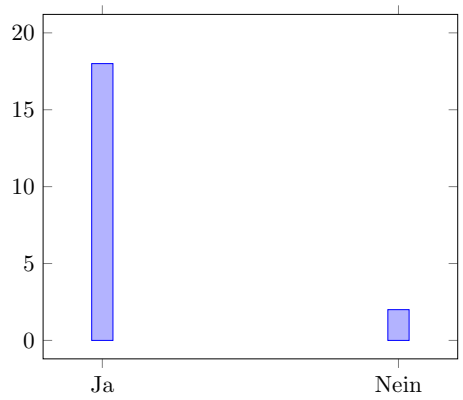


Abbildung 19: “Ich habe alle Features des Museumsexponats ausprobiert„

Abbildung 19 zeigt, dass die Mehrheit der Teilnehmer*innen alle Features des Museumsexponats ausprobiert haben. Diese sind ebenfalls in dem Programm beschrieben, wie ersichtlich ist in Abschnitt 5.0.5. Lediglich zwei Teilnehmer*innen probierten nicht alle Features aus, wobei einer dieser Teilnehmer*innen in Abbildung 18 angegeben hat, nicht gut mit dem Exponat zurechtgekommen zu sein.

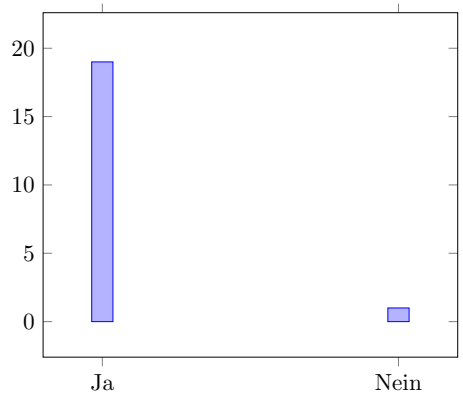


Abbildung 20: “Ich habe alle Features des Museumsexponats verstanden„

In Abbildung 20 gab die Mehrheit der Teilnehmer*innen an, alle Features des Exponats verstanden zu haben. Lediglich die Person, die bereits in den anderen Fragen Verständnisprobleme angegeben hat, verneinte auch hier die Verständlichkeit.

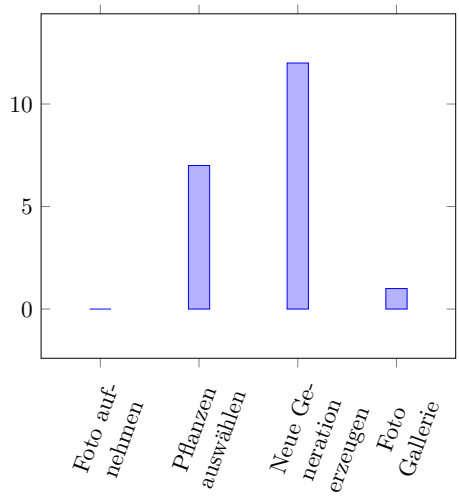


Abbildung 21: „Dieses Feature hat mir am besten gefallen“

In Abbildung 21 ist zu sehen, welches der Features des Exponats am besten bei den Teilnehmer*innen abschnitt. Interessant ist zu sehen, dass die Mehrheit die direkte Interaktion mit den Pflanzen bevorzugte.

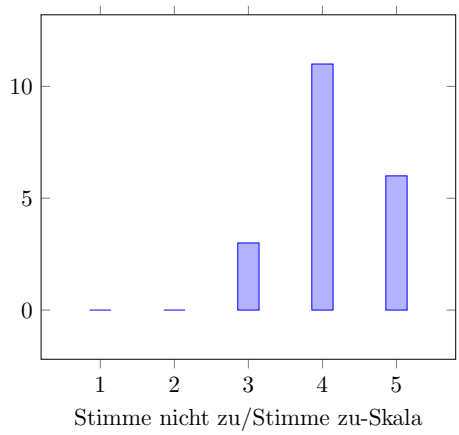


Abbildung 22: „Ich habe verstanden, wie die Evolution im Museumsexponat umgesetzt wird“

Trotz der Verständnisprobleme vereinzelter Teilnehmer*innen, gaben alle in Abbildung 22 an verstanden zu haben, wie die Evolution in dem Museumsexponat umgesetzt wird.

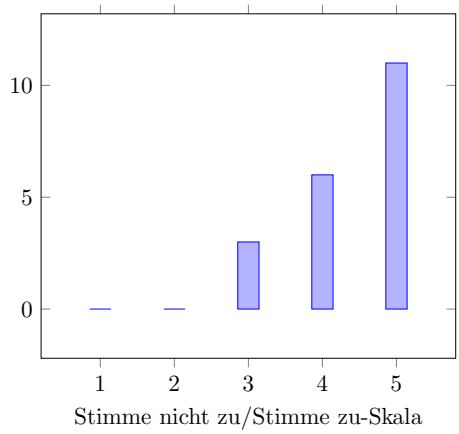


Abbildung 23: "Das Design und die Funktion des Museumsexponats sind verständlich"

Auch in Abbildung 23 gaben alle Teilnehmer*innen an, dass das Design und die Funktion des Museumsexponats verständlich sind.

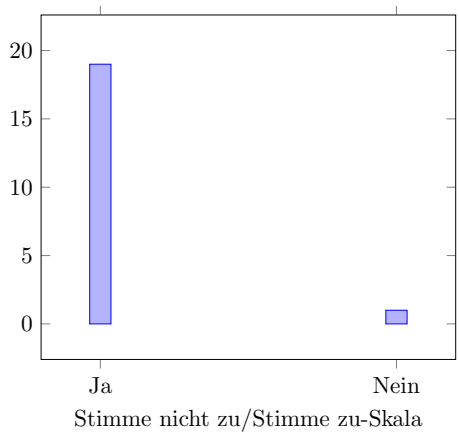


Abbildung 24: „Ich würde am Ende meines Museumrundgangs nochmal das Exponat aufsuchen, um zu gucken, wie sich die Pflanzen durch andere Besucherinteraktionen geändert haben“

Interessant ist zu sehen, dass obwohl in Abbildung 21 die Foto Features am schlechtesten abschnitten, die Mehrheit der Teilnehmer*innen nach einem Museumsrundgang trotzdem nochmal zurück zum Exponat kommen würden um zu schauen, wie sich die Generation durch andere Interaktionen verändert haben.

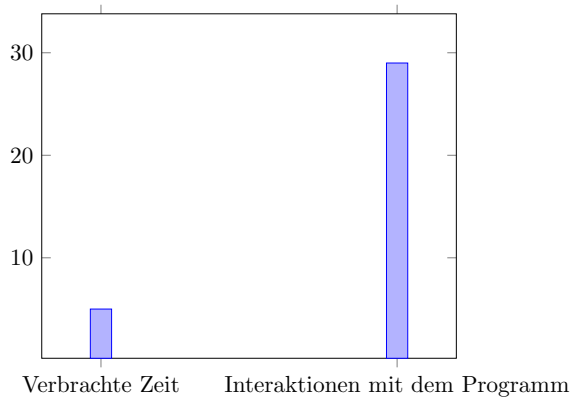


Abbildung 25: Durchschnittliche Zeit (gemessen in Minute) und Interaktionen (gemessen in Klicks) der Benutzer*innen

In Abbildung 25 sind die gemessenen Werte im Durchschnitt angeben. Ein*e Teilnehmer*in verbrachte durchschnittlich 5 Minuten an dem Exponat, wobei der niedrigste gemessene Wert bei 2 Minuten und 51 Sekunden liegt und der am höchsten gemessene Wert bei 8 Minuten und 3 Sekunden liegt. Die durchschnittliche Anzahl an Interaktionen liegt bei 29 Klicks pro Teilnehmer*in, wobei der höchste gemessene Wert bei 55 Klicks und der niedrigste Wert bei 10 Klicks liegt. Es wurden nur die Klicks gezählt, die direkt auf Pflanzen oder den Generationsbutton getätigt worden sind. In der Zeit von 6 Stunden, die das Exponat ausgestellt worden sind, sind durch 20 Teilnehmer*innen insgesamt 107 Generationen entstanden.

6.0.4 User Fotos

Im folgenden werden einige Bilder gezeigt, die von den Teilnehmer*innen im Programm aufgenommen worden sind. Insgesamt sind durch die 20 Teilnehmer*innen 44 Bilder entstanden.

Abbildung 26: Foto aufgenommen während der Studie um 10:34 Uhr

Abbildung 27: Foto aufgenommen während der Studie um 10:38 Uhr

Abbildung 28: Foto aufgenommen während der Studie um 10:39 Uhr

Abbildung 29: Foto aufgenommen während der Studie um 10:39 Uhr

Abbildung 30: Foto aufgenommen während der Studie um 10:50 Uhr

Abbildung 31: Foto aufgenommen während der Studie um 10:51 Uhr

Abbildung 32: Foto aufgenommen während der Studie um 10:55 Uhr

Abbildung 33: Foto aufgenommen während der Studie um 10:55 Uhr

Abbildung 34: Foto aufgenommen während der Studie um 11:11 Uhr

Abbildung 35: Foto aufgenommen während der Studie um 10:21 Uhr

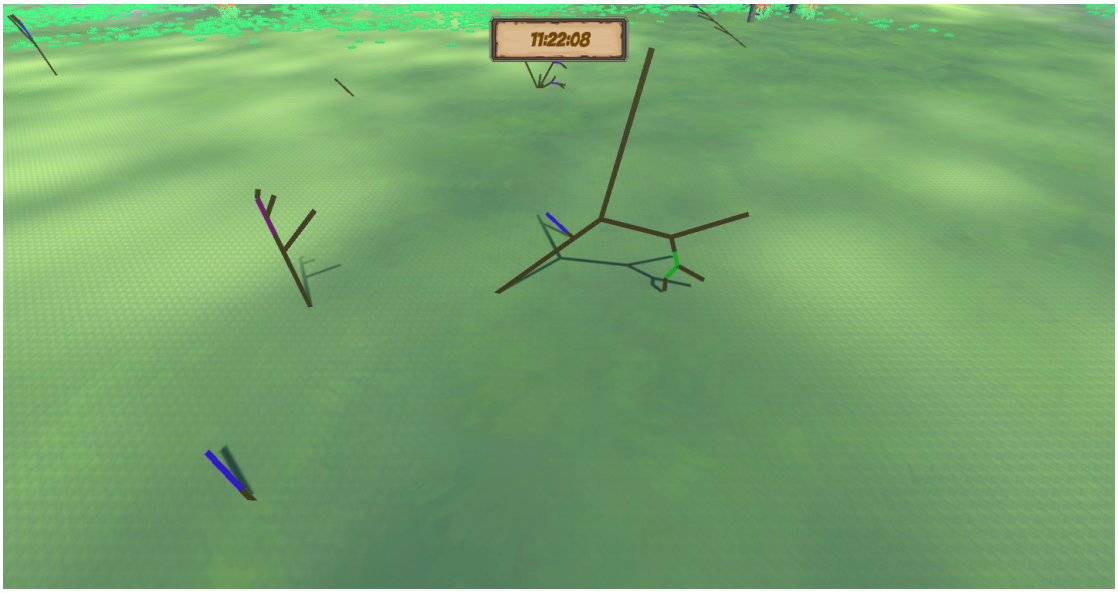


Abbildung 36: Foto aufgenommen während der Studie um 11:22 Uhr

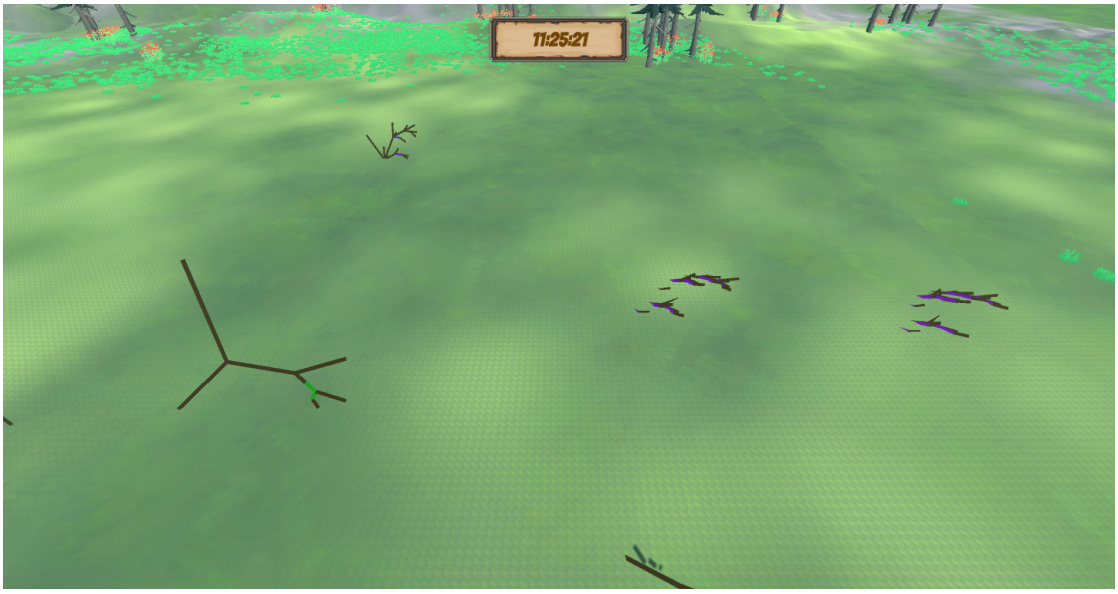


Abbildung 37: Foto aufgenommen während der Studie um 11:25 Uhr

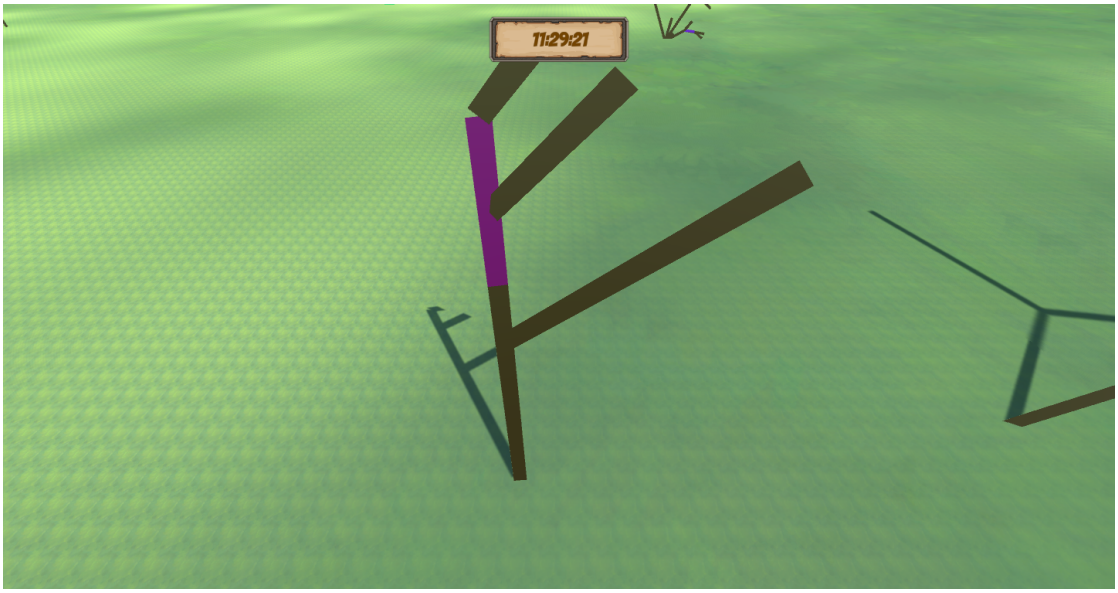


Abbildung 38: Foto aufgenommen während der Studie um 11:29 Uhr

Abbildung 39: Foto aufgenommen während der Studie um 11:33 Uhr

Abbildung 40: Foto aufgenommen während der Studie um 11:36 Uhr

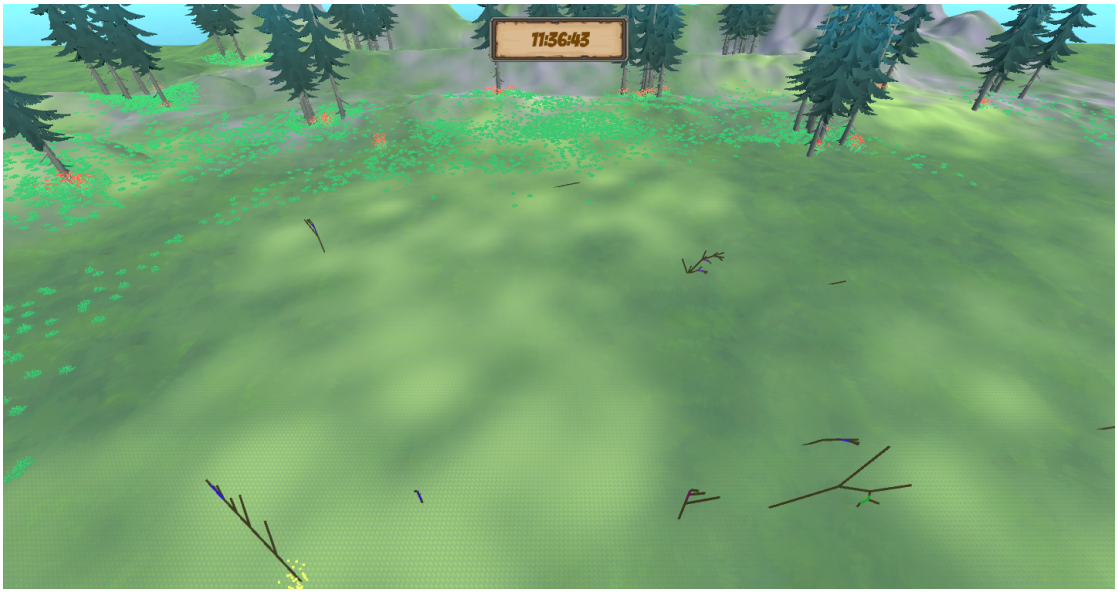


Abbildung 41: Foto aufgenommen während der Studie um 11:36 Uhr

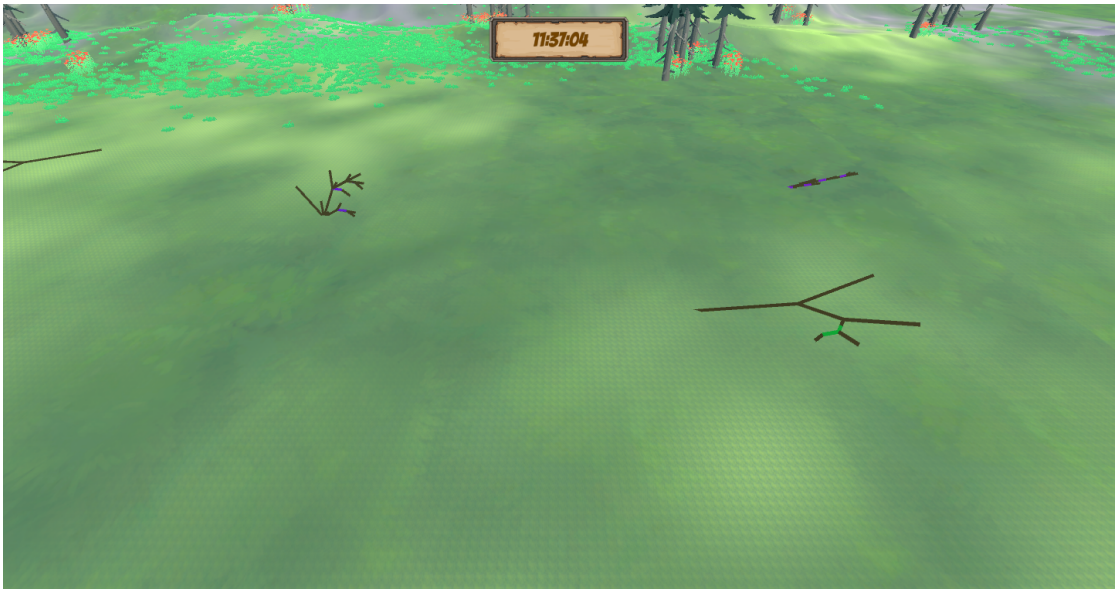


Abbildung 42: Foto aufgenommen während der Studie um 11:37 Uhr

Abbildung 43: Foto aufgenommen während der Studie um 11:37 Uhr

Abbildung 44: Foto aufgenommen während der Studie um 11:37 Uhr

Abbildung 45: Foto aufgenommen während der Studie um 11:37 Uhr

Abbildung 46: Foto aufgenommen während der Studie um 11:38 Uhr

Abbildung 47: Foto aufgenommen während der Studie um 11:46 Uhr

Abbildung 48: Foto aufgenommen während der Studie um 12:13 Uhr

Abbildung 49: Foto aufgenommen während der Studie um 12:14 Uhr

Abbildung 50: Foto aufgenommen während der Studie um 12:19 Uhr

Abbildung 51: Foto aufgenommen während der Studie um 12:43 Uhr

Abbildung 52: Foto aufgenommen während der Studie um 12:49 Uhr

Abbildung 53: Foto aufgenommen während der Studie um 12:50 Uhr

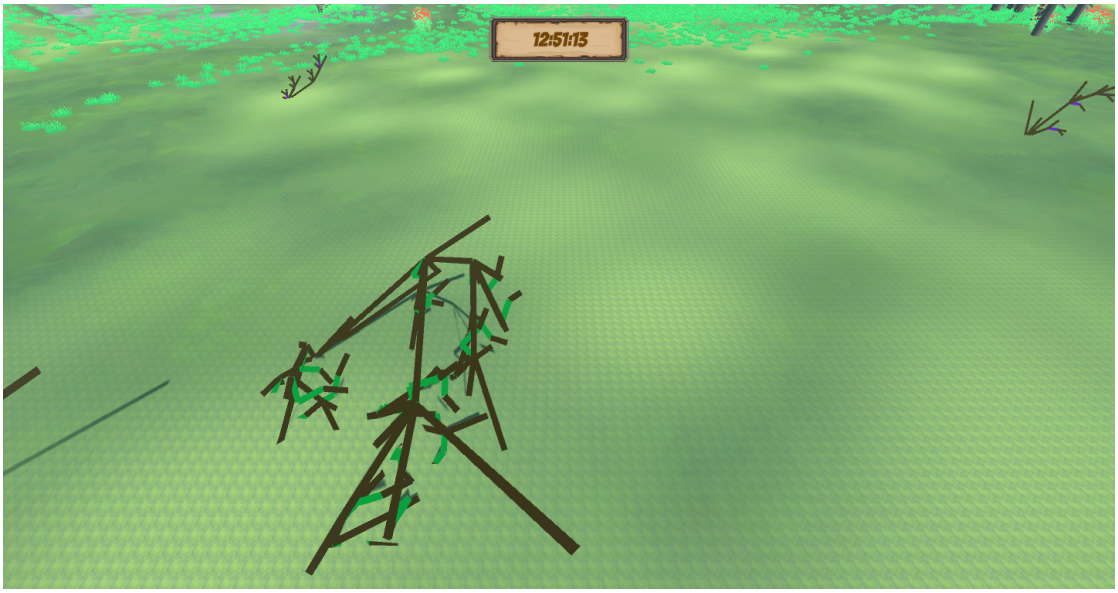


Abbildung 54: Foto aufgenommen während der Studie um 12:51 Uhr

Abbildung 55: Foto aufgenommen während der Studie um 12:51 Uhr

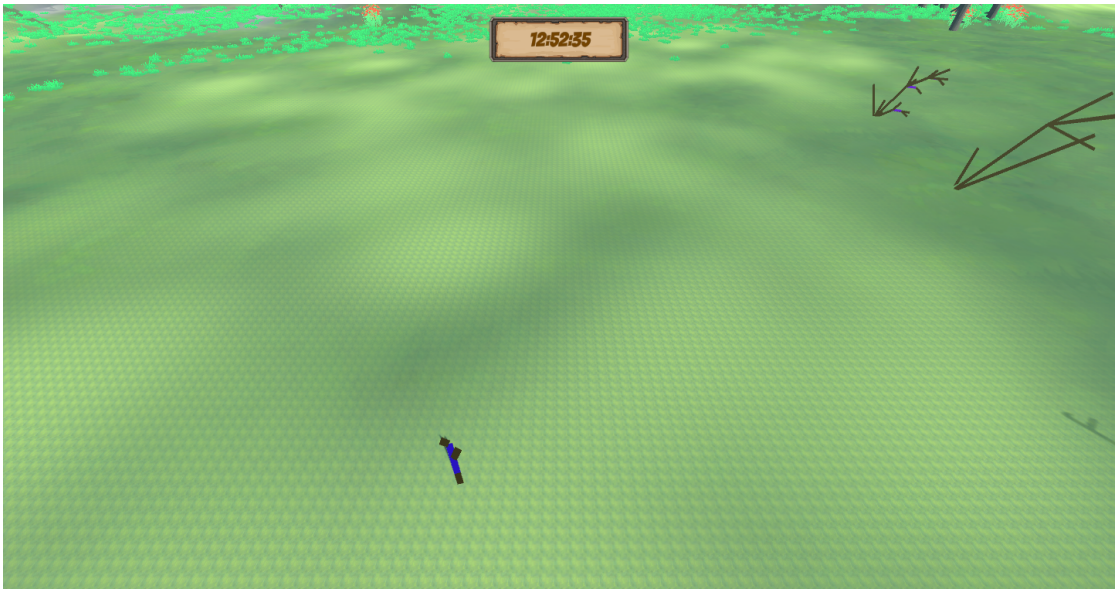


Abbildung 56: Foto aufgenommen während der Studie um 12:52 Uhr

Abbildung 57: Foto aufgenommen während der Studie um 12:57 Uhr

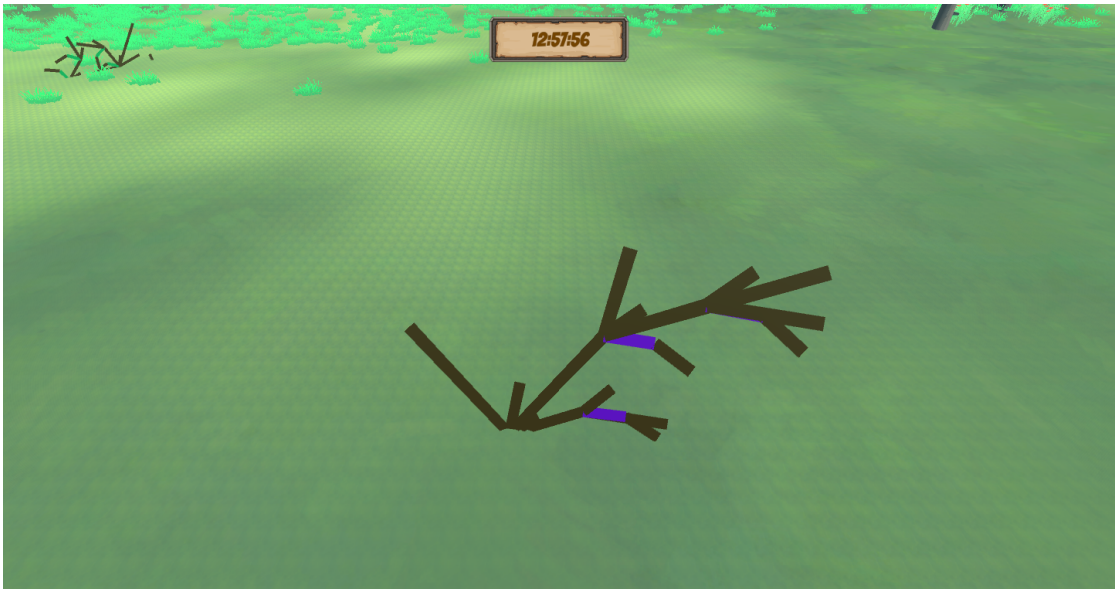


Abbildung 58: Foto aufgenommen während der Studie um 12:57 Uhr

Abbildung 59: Foto aufgenommen während der Studie um 12:59 Uhr

Abbildung 60: Foto aufgenommen während der Studie um 13:05 Uhr

Abbildung 61: Foto aufgenommen während der Studie um 13:05 Uhr

Abbildung 62: Foto aufgenommen während der Studie um 13:09 Uhr

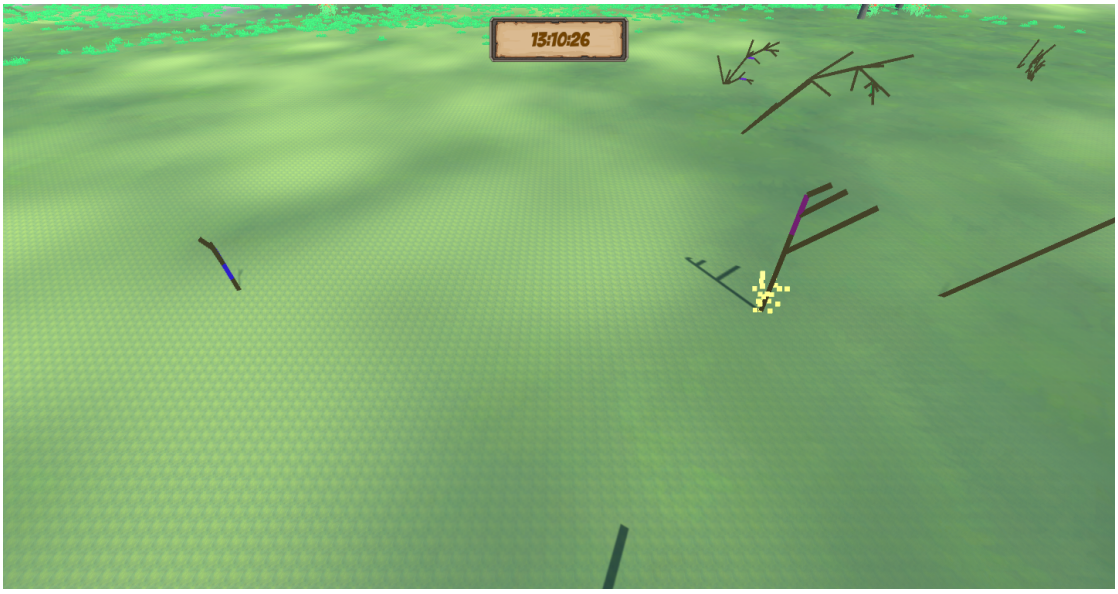


Abbildung 63: Foto aufgenommen während der Studie um 13:10 Uhr

Abbildung 64: Foto aufgenommen während der Studie um 13:13 Uhr

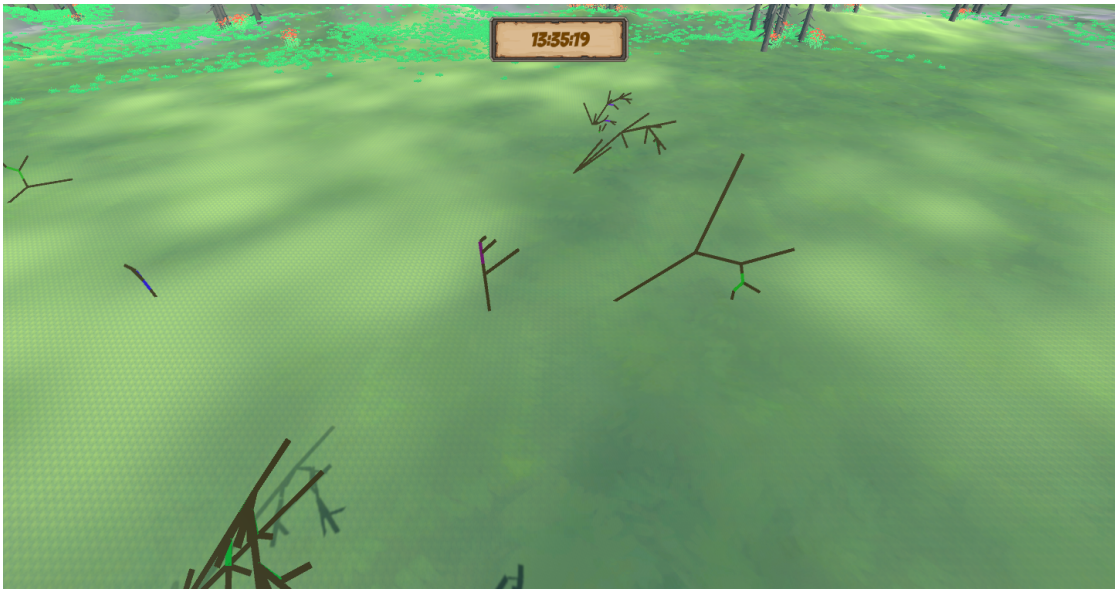


Abbildung 65: Foto aufgenommen während der Studie um 13:35 Uhr

Abbildung 66: Foto aufgenommen während der Studie um 13:42 Uhr

Abbildung 67: Foto aufgenommen während der Studie um 13:42 Uhr

Abbildung 68: Foto aufgenommen während der Studie um 13:42 Uhr

Abbildung 69: Foto aufgenommen während der Studie um 13:50 Uhr

6.0.5 Auswertung

In Anbetracht der Ergbenisse, welche der User Study zu entnehmen sind, kann festgehalten werden, dass das System noch Defizite aufweist. Das größte Problem, welches durch die Fragen erkannt werden konnte, ist die Verständlichkeit des Programms. Für Teilnehmer*innen, die kaum oder keine Berührungspunkte mit dem Thema Evolution hatten fiel der Umgang mit dem Exponat und das Verständnis der Darstellung der Evolution schwerer. Zu bedenken wäre, dass Nutzer in einem Museumssetting noch mehr Informationen (Infotafeln, Artikel etc.) zur Verfügung hätten.

Interessant war, dass die Teilnehmer*innen mehr in die Interaktion mit dem Exponat investiert haben, als ursprünglich angedacht war. Bei einer Gesamtanzahl von 107 Generationen bei 20 Teilnehmer*innne, hat jede*r ca. 5 Generationen generiert. Obwohl ein Großteil der Teilnehmer*innen in Abbildung 21 die Foto Galerie und das Aufnehmen von Fotos nicht als Feature wählten, das ihnen am besten gefiel, hat jede*r mindestens ein Foto geschossen. Aus den Timestamps der Bilder aus Abschnitt 6.0.4 lässt sich außerdem entnehmen, dass einige Teilnehmer*innen innerhalb kürzester Zeit mehrere Bilder aufgenommen haben. Die Foto Funktion wurde also dennoch reichlich genutzt und könnte für zukünftige Veränderungen noch erweitert werden.

Aufschlussreich war unter anderem, dass die Teilnehmer*innen nicht unbedingt die Pflanzen favorisiert haben, die echten Pflanzen ähneln, sondern auch skurrile Ergebnisse wie in Abbildung 54 und Abbildung 59 bevorzugten. Eine genaue Erklärung wieso die Teilnehmer*innen diese bevorzugten, haben sie nicht angegeben. Ebenefalls ist zu erkennen, dass sich die Blütenfarbe grün durchgesetzt hat, obwohl die Teilnehmer*innen bei der Form der Pflanzen eher zu skurrilen Modellen tendierten.

Abbildung 69 ist das letzt Bild, auf der der Großteil der Population abgebildet ist. Hier kann man deutlich sehen, dass die meisten Pflanzen ihre Winkelneigung nach rechts haben und insgesamt nach rechts wachsen. Daraus kann man schließen, dass sich die Gene eben dieser Pflanzen und ihrer Eltern über die vielen Generationen gut durchgesetzt haben. Auf den Abbildungen in Abschnitt 6.0.4 ist außerdem zu erkennen, dass einige Pflanzen sich über mehrere Generationen durchsetzen konnten. Die Pflanze die in Abbildung 26 zu sehen ist, ist ebenfalls in Abbildung 69 zu sehen. Diese hatte durch die Interaktionen der Teilnehmer*innen immer wieder einen der höchsten Fitnesscores. Einige Teilnehmer*innen klickten einige Pflanzen öfter als einmal an, so dass diese gleich mehrere Punkte auf ihren Fitnesscore gutgeschrieben bekommen haben. Für zukünftige Veränderungen könnte dies bedeuten, dass man ebenfalls wie beim Generationsbutton einen Cooldown für den Fitnesscore einfügt, um mehr Vielfalt und Veränderung zu fördern.

7 Fazit und Ausblick

Evolution und die aus ihren abgeleiteten Systeme und Konzepte sind seit Jahren eine wichtige Grundlage in Computerprogrammen und haben schon in einigen Studien Verwendung gefunden.

Ziel der vorliegenden Studie war es, ein interaktives Museumsexponat mithilfe des L-Systems zu konzipieren, das Evolution naturgetreu darstellt und trotzdem für Laien verständlich ist. Um sich dem Thema der simulierten Evolution anzunähern, habe ich mir zunächst einige Studien angeschaut, welche Ansätze benutzen, die aus diesem Bereich stammen. Durch diese Recherche konnten die Grundlagen festgelegt werden, welche ich für das Museumsexponat benötigt habe. Diese Grundlagen setzten sich zusammen aus dem L-System und einem genetischen Algorithmus. Für das L-System wurde ein Alphabet implementiert, welches die Umschreibregeln festlegt. Diese wurden schließlich für die Darstellung der Pflanzen verwendet. Der genetische Algorithmus dient dazu den Evolutionsprozess darzustellen und das Vererben der Merkmale zu ermöglichen. Dieser Algorithmus initialisiert die Population, wertet diese aus und legt die Vererbung fest. Aus diesen beiden Ansätzen und spielerischen Elementen habe ich das Museumsexponat programmiert. Um die Qualität der implementierten Evolution und die Usability des Exponats zu testen, habe ich eine Evaluation durchgeführt.

Aufgrund der geringen Anzahl an Teilnehmenden an der Evaluation in Abschnitt 6, können keine allgemeingültigen Schlussfolgerungen zu der Usability des Museumsexponats gezogen werden. Dennoch erlaubt die Evaluation einen Einschätzung davon, wie das Exponat aufgenommen wurde. Ein Großteil der Teilnehmer*innen kamen mit dem System gut zurecht, besonders die, die bereits Berührungspunkte mit dem Bereich der Evolution hatten. Der Evolutionsprozess des Exponats welcher in der Evaluation über mehrere Stunden lief zeigt auf, dass die Kombination des L-Systems mit dem genetischen Algorithmus gut funktioniert. Auch die Vererbung der vielen verschiedenen Gene wie Teilstrings, Blütenfarbe, Winkelwerte funktioniert gut, was sich besonders aus den Bildern aus Abschnitt 6.0.4 entnehmen lässt. Die direkte Interkation mit den Pflanzen wurde von den Teilnehmer*innen favorisiert. Trotzdem kann man durch die Anzahl der Bilder in Abschnitt 6.0.4 darauf schließen, dass auch die Veränderung der Generationen über den Evolutionszeitraum interessant für Benutzer*innen ist.

Das Museumsexponat besitzt bereits gute Ansätze und setzt die Evolution gut um. In Zukunft könnten noch weitere Features hinzugefügt werden. Für eine weitere Forschungsarbeit wäre es interessant, das Exponat und seine Funktionen zu erweitern. Zusätzlich zu dem Klicken als Einfluss auf die Fitness der Pflanzen könnten noch weitere naturgetreue Features hinzugefügt werden, die den Evolutionsprozess beeinflussen.

Literatur

[1] Robert Collins and David Jefferson. Antfarm: Towards simulated evolution. 06 2001.

[2] John J. Holland. Genetic algorithms. *Scientific American*, 267(1), 1992.

[3] Aristid Lindenmayer and Przemyslaw Prusinkiewicz. *The Algorithmic Beauty of Plants*. Springer-Verlag, 1990.

[4] Benjamin Mark, Tudor Berechet, Tobias Mahlmann, and Julian Togelius. Procedural generation of 3d caves for games on the gpu. 2015. Foundations of Digital Games ; Conference date: 22-06-2015.

[5] Karl J. Niklas. Computer-simulated plant evolution. *Scientific American*, 254(3), 1986.

[6] Gabriela Ochoa. *Parallel Problem Solving from Nature — PPSN V*. Springer Berlin Heidelberg, 1998.

[7] Daniel Shiffman. *The Nature of Code*. Daniel Shiffman, 2012.