HOCHSCHULE
DER MEDIEN

Neuronale Netze in der Charakteranimation

EDV-Nr.: 253504A Aktuelle Themen

Dozent: Prof. Dr. Andreas Koch

Student:
Daniel Dreher
dd058@hdm-stuttgart.de
Matr.-Nr.: 36407

Inhaltsverzeichnis

Einleitung

Die Welt befindet sich gerade inmitten einer technischen Revolution durch künstliche Intelligenz (KI). Beinahe täglich liest man von neuen Anwendungsgebieten, wissenschaftlichen Durchbrüchen und gebrochenen menschlichen Rekorden in der Branche. Vor nicht allzu langer Zeit schien die KI noch in weiter Ferne. Mittlerweile nutzt sie ein Großteil der Menschheit unterbewusst, sei es durch Smartphones, Suchmaschinen oder sogar Fahrassistenzsystemen im Auto. Die KI ist bereits im Alltag angekommen und Firmen müssen sich die Frage stellen wie sie eingesetzt werden kann, damit sie den Anschluss an die Konkurrenz nicht verpassen. Gerade im Bereich der Unterhaltungselektronik herrscht ein extremer Wettbewerb. Gleichzeitig ist die Spiele-Industrie aufgrund ihrer Schnelllebigkeit häufig Vorreiter und frühzeitiger Anwender neuer Technologien. Wie kann die KI in diesem Umfeld hilfreich sein?

Im Folgenden soll der Einsatz von KI im Bereich der Echtzeit-Charakteranimation beschrieben werden.

1 Stand der Technik

Traditionell werden bis zur finalen Darstellung eines digitalen Modells immer drei Arbeitsschritte durchlaufen: Das eigentliche Modellieren, das Einfügen eines Skeletts als Vorbereitung für die Animation (Rigging) und letztendlich die Erstellung der benötigten Animationen.

1.1 Modellierung

Ein digitales Modell, das für die Animation in Echtzeit bestimmt ist, besteht regulär aus einer Vielzahl von Vierecken bzw. Dreiecken (Abbildung 1). Basierend auf einem Konzept werden jegliche dargestellten Gegenstände und Charaktere mit Hilfe von spezialisierter Modellierungs-Software erstellt. Während Festkörper, je nach Anforderungen des Projekts, teilweise automatisiert generiert werden können, z.B. durch Photogrammetrie, ist dies bei Charakteren meistens nicht der Fall. Organische Elemente müssen meist manuell modelliert werden. Sculpting-Software, wie z.B. ZBrush von Pixologic, erlaubt es den Künstlern hier ähnlich wie beim Modellieren mit Ton oder Lehm vorzugehen. Die erstellten Modelle sind zu diesem Zeitpunkt meist sehr hoch aufgelöst und müssen anschließend reduziert werden. Hierfür kommen Softwarepakete wie Maya und 3dsmax von Autodesk zum Einsatz, die gleichzeitig auch Werkzeuge zum Rigging und Animieren bieten.

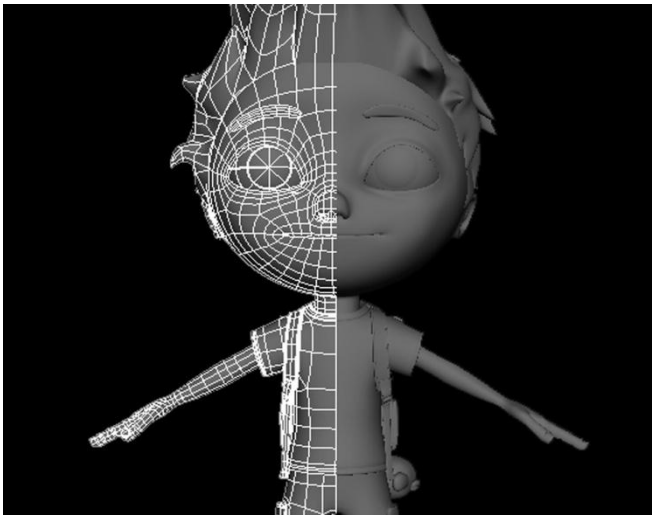

Abbildung 1: Digitales Charaktermodell

1.2 Rigging

Da das erstellte Modell letztendlich nur eine Ansammlung von Polygonen ist, kann es nicht ohne weiteres animiert werden. Man benötigt zusätzlich noch weitere Daten um festlegen zu können, welche Polygone bei einer Animation wie verformt werden sollen. Bei modernen Animations-Pipelines wird für diesen Zweck das sogenannte Rigging ausgeführt. Bei diesem Prozess wird ein digitales Skelett für das Modell erstellt. Gleichzeitig wird für jeden Punkt des Modells festgelegt wie stark er von den einzelnen Gelenken und Knochen beeinflusst wird.

1.3 Animation

Nach dem Rigging ist das Modell für die Animation vorbereitet. Für die Umsetzung der Animation gibt es nach aktuellem Stand der Technik zwei Möglichkeiten: Erstellen von Frame-by-Frame-Animationen oder die Animation durch Motion Capture.

1.3.1 Frame-by-Frame-Animation

Die Frame-by-Frame-Animationen basieren auf einer traditionellen Technik, die dem Erstellen eines Daumenkinos ähnelt. Für jede Animation werden manuell Keyframes festgelegt. Ein Keyframe beschreibt eine Änderung der Knochen- bzw. Gelenkstellungen in einer Animation zu einem Zeitpunkt in der Animation. Beim Abspielen der Animation werden die Transformationen des Skeletts über die Zeit entsprechend der Keyframes interpoliert.

Trotz des hohen Aufwands und der Entwicklung modernerer Techniken, wie z.B. Animation mit Motion Capture, sind Frame-by-Frame-Animationen immer noch weit verbreitet, da der Animator hier die volle Kontrolle über die dargestellten Bewegungen hat. Speziell für stilisierte Bewegungsabläufe oder fiktive Figuren ist diese Technik immer noch relevant.

1.3.2 Motion Capture

Qualitativ hochwertige, realistische Bewegungen manuell Frame-by-Frame zu erstellen ist nicht nur zeitaufwändig, sondern benötigt zudem auch noch domänenspezifisches Wissen. Der Animator muss einschätzen können, wie sich eine Kreatur bewegen kann, muss wissen wie genau die Bewegungsabläufe eines Charakters aussehen und welche Details der Bewegungen den Realismus ausmachen.

Eine vergleichsweise einfachere Methode um realistische Animationen zu erzeugen, vor allem wenn es um menschenähnliche Bewegungsabläufe geht, ist die Verwendung einer Motion-Capture-Technologie. Kameras nehmen hierbei Bewegungen eines Menschen aus mehreren Blickwinkeln auf, die dann auf ein digitales Modell übertragen werden

können. Die aufgenommenen Daten müssen anschließend nur noch von etwaigem Rauschen bereinigt werden und können dann für die Animation verwendet werden.

Problematisch hierbei ist jedoch die Verwaltung der aufgenommenen Animationen. Man kann nicht eindeutig identifizieren wann eine bestimmte Animation beginnt und endet. Zusätzlich fällt die verständliche Benennung komplexer Animationsabläufe schwer. Die Einhaltung und Regelung von Standards nimmt mehr Zeit in Anspruch.

1.4 Verwendung der Animationen

Während im Offline Rendering die vorbereiteten Modelle nun bereits verwendet werden könnten, ist dies im Falle von Echtzeit-Simulationen nicht der Fall. Dargestellte Inhalte und Bewegungen sind hier oft interaktiv und hängen von den Eingaben des Benutzers oder dem momentanen Zustand der Simulation ab. Um die benötigten Animationen darstellen zu können ist eine weitere Schicht zur Berechnung und Interpolation der finalen Pose eines Modells nötig.

1.4.1 Animationslogik

Aktuelle Spiele-Engines nutzen für die Berechnung der angezeigten Animation eine Mischung aus sogenannten State Machines und Blend Trees. Eine State Machine bildet den momentanen Zustand eines Charakters abhängig von der Benutzereingabe und dem Zustand der Simulation ab. So kann eine Figur beispielsweise gehen, rennen, still stehen, springen oder fallen. Zu jedem dieser Zustände gehört eine bestimmte Animation, die nach dem zuvor beschriebenen Prozess aufbereitet wurde (Abbildung 2).

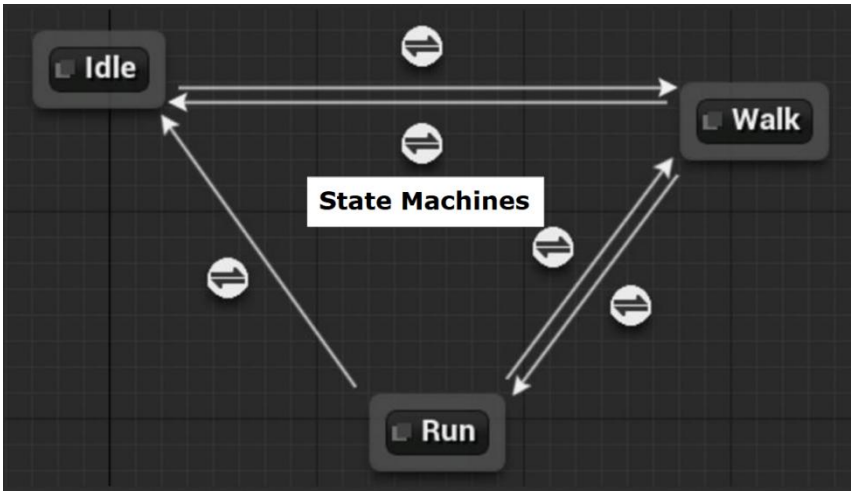

Abbildung 2: Finite State Machine

Zusätzlich ist für eine realistische Animation ein Mechanismus nötig, der die Animationsübergänge bei einem Wechsel des States in der State Machine berechnet. Für diesen Zweck werden Blend Trees verwendet (Abbildung 3). Sie berechnen eine Interpolation der Animationsdaten der einzelnen States und können wiederum State Machines enthalten.

Die gesamte Animationslogik wird manuell von Programmierern erstellt und über den Lauf des Projektes erweitert und angepasst. Es entsteht ein komplexes Netz an verschachtelten State Machines und Blend Trees, dessen Resultat am Ende die finale Pose der Figur widerspiegelt. Folglich müssen die Statemachines bei der Einbindung neuer Animationen regelmäßig verändert und gewartet werden. Bei großen Projekten kann diese Arbeit viel Zeit in Anspruch nehmen.

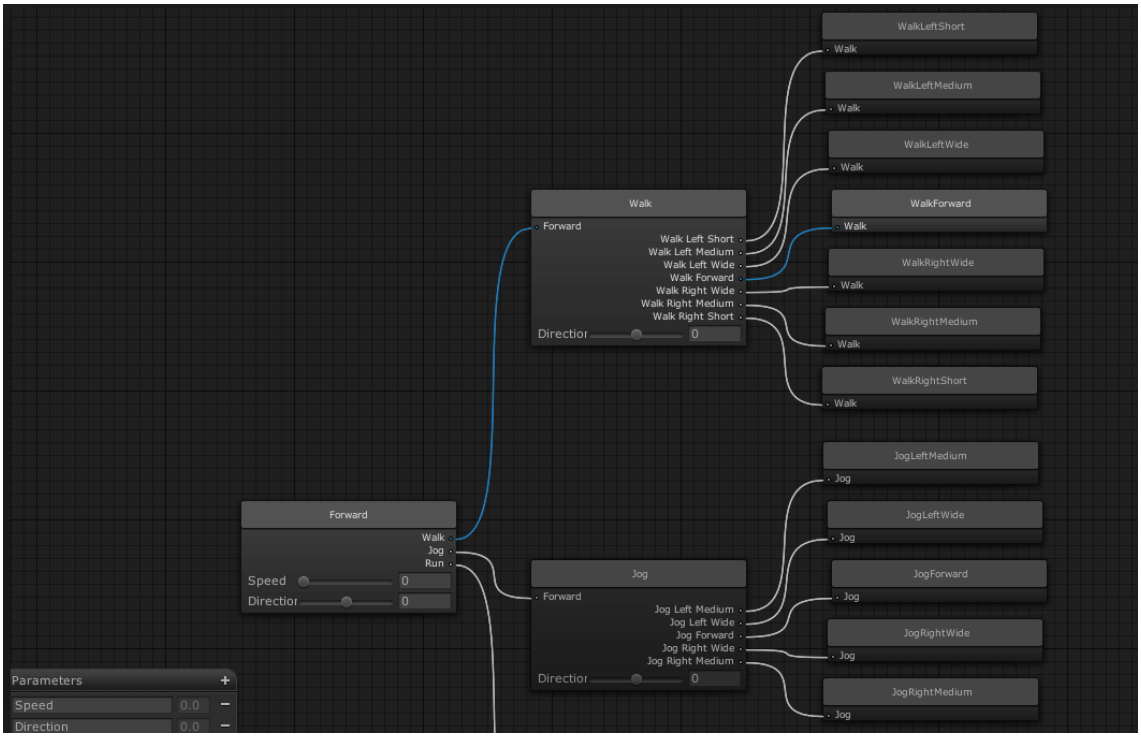

Abbildung 3: Parametrisierte Blend Trees

2 Vereinfachung durch KI

Viele der beschriebenen Probleme sind ein Resultat der Datenkomplexität von Animationen und deren Echtzeit-Evaluation. Neuronale Netze leben von einer großen Menge an Daten und es hat sich herausgestellt, dass sie sich auch für dieses Anwendungsgebiet gut eignen. Hierfür muss die beschriebene Animationslogik in ein datengetriebenes Problem transformiert werden.

2.1 Neuronale Netze zur Pose-Evaluation

Die Motion-Capture-Daten werden mit numerischen Labels versehen, wie es bei Supervised Learning von neuronalen Netzen üblichen ist. So können die aufgenommenen Animationen z.B. Labels für das Geschlecht der dargestellten Figur, für die Bewegungsgeschwindigkeit, die Bewegungsrichtung oder die Rotation haben. Gleichzeitig werden die State Machines so angepasst, dass sie nicht mehr über Blend Trees die finale Pose der Figur evaluieren, sondern lediglich eine Kombination an Labels zusammenstellen, aus denen die finale Position errechnet werden soll. (Abbildung 4)

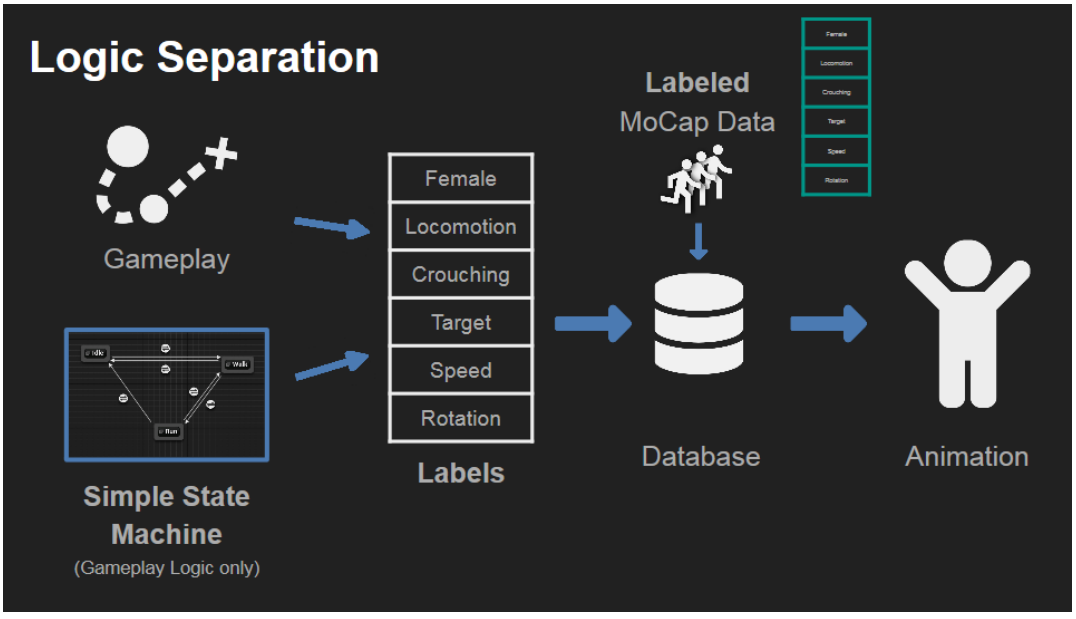

Abbildung 4: Labeling der Animationen

Man entkoppelt damit die Logik, die die Eingaben des Spielers und die damit verbundenen Regeln verarbeitet von der dargestellten Animation. Basierend auf den berechneten Feature-Vektoren und den dazugehörigen gelabelten Animationsdaten des Motion Captures, die letztendlich auch numerische Vektoren darstellen, die die Transformation jedes Gelenks beschreiben, ist es nun möglich ein Regressions-Netz zu trainieren. Ein sogenanntes Phase Functioned Neural Network liefert hier stabile Ergebnisse (Abbildung 5). Die Gewichte des neuronalen Netzes werden in dieser Architektur durch eine sogenannte Phase Function moduliert. Für jede Phase wird ein eigener Satz an Gewichten gelernt. Die Phasen werden mit der Zeit der Reihe nach durchlaufen.

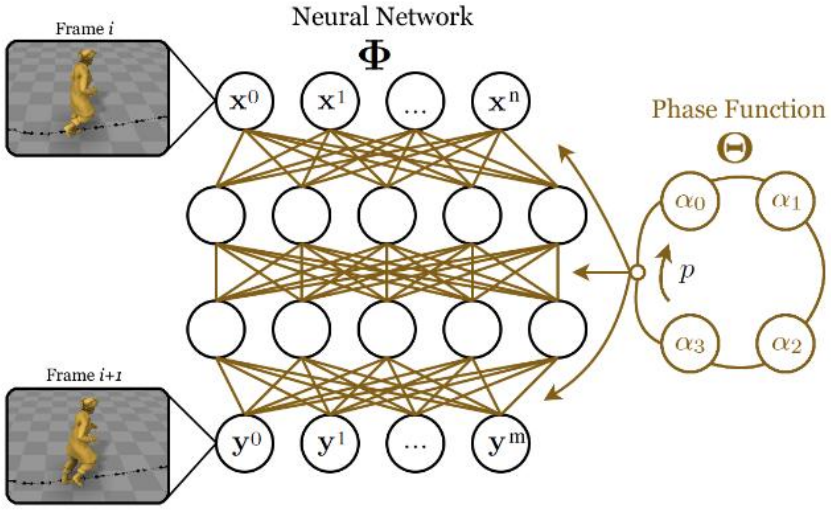

Abbildung 5: Phase Functioned Neural Network

Eingabe des neuronalen Netzes ist ein Feature-Vektor der den Input des Benutzers mit den Simulationsregeln kombiniert. Als Trainingsdaten dienen die gelabelten Daten des Motion-Capture-Prozesses. Die Ausgabe des neuronalen Netzes ist die erwartete Pose der Figur als numerischer Vektor. Die Fehlerfunktion für das Training berechnet beispielsweise die euklidische Distanz zwischen den Trainings- und den Ausgabe-Vektoren. Zusätzlich bietet es sich an die Differenz der vom Benutzer gewollten Trajektorie und der Trajektorie der Animation auszuwerten um eine möglichst passende Animation auszuwählen.

Im Betrieb des neuronalen Netzes wird die Ausgabe des Netzes mit der vorherigen Pose interpoliert. In Kombination mit gängigen Animationstechniken, wie z.B. inverser Kinematik, entstehen hierdurch flüssige Animationsübergänge, die glaubhaft mit der Umwelt der Simulation interagieren.

3 Bewertung

Viele der angesprochenen Probleme, die beim Arbeiten mit Computeranimationen entstehen, werden durch die Verwendung von neuronalen Netze umgangen.

Die Verwaltung von einzelnen Animationen und die manuelle Programmierung der Animationslogik entfallen. Im Falle von sich häufig ändernden Anforderungen an die benötigten Animationen vereinfacht ein neuronales Netz den Aufwand ungemein, da bereits komplexer Code nicht noch weiter mit Spezialfällen verkompliziert werden muss. Das Hinzufügen zusätzlicher Animationen benötigt lediglich mehr Motion-Capture-Daten, Einpflegen neuer Labels in die Simulationslogik des Programms und ein erneutes Training des Regressors.

Ein weiterer Vorteil neuronaler Netze sind die gesenkten Hardware-Anforderungen im Betrieb. Die Motion-Capture-Daten müssen nicht auf die Hardware des Nutzers übertragen werden, da sie bereits implizit in dem gelernten Modell des neuronalen Netzes vorhanden sind. Sie werden stark komprimiert. 1,5 GB an Daten werden somit auf ungefähr 10-100 MB reduziert. Gleichzeitig lässt sich das neuronale Netz zur Laufzeit innerhalb weniger Millisekunden per Frame auf der CPU evaluieren.

Der Einsatz neuronaler Netze bringt jedoch nicht nur Vorteile mit sich. Ein Nachteil von neuronalen Netzen ist die benötigte Zeit um das Modell anzutrainieren. Die Ergebnisse sind nicht sofort sichtbar und können erst nach Stunden beurteilt werden. Da die herkömmlich Art der Animationsaufbereitung jedoch auch eine gewisse Zeit benötigt und Kosten verursacht wird dieser Aufwand wieder relativiert.

Problematisch ist auch die Kontrolle der abgespielten Animationen. Das neuronale Netz ist vorerst eine Blackbox und kann nicht gezielt beeinflusst werden. Möchte man nun also eine Reihe festgelegter Animationen abspielen, z.B. bei Zwischensequenzen, ist eine hybride Lösung mit Rückfall auf herkömmliche Animationstechniken anzuraten.

Letztendlich fordert diese neue Animationstechnik auch eine vollkommen neue Ausbildung für die Animatoren. Anstatt sich mit der Analyse und Replikation realistischer Bewegungen auseinanderzusetzen ist nun Expertise im Bereich Machine und Deep Learning gefragt.

4 Zusammenfassung und Ausblick

Im Vergleich zu herkömmlichen Animationsverfahren ermöglichen Neuronale Netze die Umsetzung realistischer Animationen in Echtzeit mit relativ wenig Aufwand. Wo bisher viele manuelle Arbeitsschritte nötig waren um die Animationen aufzubereiten, zu verwalten und in die Programmlogik einzupflegen, basieren neuronale Netze lediglich auf einer großen Menge an gelabelten Daten. Diese können auf einfach Art und Weise durch Motion Capturing produziert werden.

Obwohl neuronale Netze viele Vorteile bieten sind sie dennoch keine endgültige Lösung. Die Hardware-Anforderungen und der benötigte Aufwand zur Implementierung der Animationen werden zwar verringert, allerdings existieren dennoch Grenzfälle, in denen herkömmliche Animationsverfahren besser abschneiden. Das Netz ist eine Blackbox und kann nicht ohne weiteres gezielt zu einer bestimmten Animation geführt werden. Zudem lassen sich stilisierte Animationen beispielsweise zum Zeitpunkt der Recherche noch nicht mit neuronalen Netzen generieren. Das beschriebene Verfahren basiert auf Motion Capturing. Folglich können fiktiven Kreaturen auf diese Weise noch nicht animiert werden. In naher Zukunft wird KI klassische Animatoren also nicht ersetzen, sondern lediglich ergänzen. Dennoch lohnt es sich für Animatoren sich mit KI auseinanderzusetzen. Neuronale Netze kommen schon heute in großen Titeln wie z.B. Ubisofts „For Honor“ oder Naughty Dogs „The Last Of Us 2“ erfolgreich zum Einsatz und der Anteil von KI animierten Inhalten wird voraussichtlich in Zukunft noch weiter steigen.

Glossar

Photogrammetrie: Photogrammetrie ist ein Verfahren um Objekte mittels Fotografie als digitales Modell abzubilden.

CGI: Computer Generated Imagery ist ein Oberbegriff für alle am Computer erstellten Beiträge zu Bildern jeglicher Art, z.B. Kunst, Videospiele, Filme

Engine: Als Engine wird der Applikationskern eines Videospiels bezeichnet, der für grundlegende Aufgaben zuständig ist, wie beispielsweise die Verwaltung von Daten, das Berechnen des angezeigten Bildes oder das Abspielen von Video- und Audioaufnahmen.

Rigging: Das Versehen eines digitalen Modells mit einem Skelett zur Animation.

State Machine: Ein Zustandsautomat bildet ein System von Zuständen und Zustandsübergängen ab.

State: Ein Zustand in einem Zustandsautomat

Blend Tree: Eine parameterisierte Interpolation mehrerer Animationen.

Frame: Ein Update-Zyklus eines Videospiels. Videospiele streben in der Regel mindestens 30-60 Frames pro Sekunde an um ein flüssiges Bild darstellen zu können.

Supervised Learning: Ein Verfahren, das zum Anlernen von neuronalen Netzen eingesetzt wird. Beim Training eines Netzes mit überwachtem Lernen wird dem Netz zu einer Eingabe eine gelabelte Ausgabe vorgegeben.

Feature-Vektor: Eine Ansammlung von Eingabedaten für ein neuronales Netz. Die Eingabe wird als numerischer Vektor umgesetzt, wie aus der Mathematik bekannt.

Regressor: Ein Machine Learning Modell, das die Trainingsdaten so gut wie möglich annähert.

Inverse Kinematik: Ein Verfahren in der digitalen Animationstechnik, das dem Animator ermöglicht die Position eines bestimmten Gelenks vorzugeben, woraufhin die Stellung der damit verbundenen Gelenke angepasst wird. Auf diese Weise kann zum Beispiel ein Fuß direkt auf den Boden gesetzt werden, obwohl die Animation den Fuß an einer anderen Stelle, über dem Boden schwebend, vorsieht.

Abbildungsverzeichnis

Quellenverzeichnis

Holden et al. (2017): Phase Functioned Neural Networks for Character Control.

Clavet (2016): Motion Matching – Road to Next-Gen Animation.
https://www.gdcvault.com/play/1023280/Motion-Matching-and-The-Road
(Datum des Zugriffs: 17. Februar 2019).

Zadziuk (2016): Motion Matching – The Future of Gameplay Animation… Today
https://www.gdcvault.com/play/1023115/Animation-Bootcamp-Motion-Matching-The
(Datum des Zugriffs: 17. Februar 2019).

Buttner (2015): Motion Matching – The Road to Next Gen Animation
https://www.youtube.com/watch?v=z_wpgHFSWss
(Datum des Zugriffs: 17. Februar 2019).

Abbildung 1: Digitales Charaktermodell
https://3dtotal.com/galleries/fbuy/toon-character-model-wireframe-by-matthewcharles
(Datum des Zugriffs: 17. Februar 2019).

Abbildung 2: Finite State Machine
https://twvideo01.ubm-us.net/o1/vault/gdc2016/Presentations/Clavet_Simon_MotionMatching.pdf
(Datum des Zugriffs: 17. Februar 2019)

Abbildung 3: Parametrisierte Blend Trees
https://forum.unity.com/proxy.php?image=http%3A%2F%2Fi.imgur.com%2F0KUbb.png&hash=2f3fda02fa61a4c2379b1a9a4acc511e
(Datum des Zugriffs: 17. Februar 2019)

Abbildung 5: Phase Functioned Neural Network
Holden et al. (2017)