Procedural Generation in Indie Games: Techniken und Herausforderungen

Felix Kargl
MIT Mobile
FH St.Pölten
St.Pölten Niederösterreich
Österreich
it241507@fhstp.ac.at

ABSTRACT

Dieser Artikel untersucht die Technologien und Herausforderungen der Procedural Generation (PG) in der Spieleentwicklung. Neben Algorithmen wie Pseudozufallsgenerierung, Perlin Noise, fraktalen Algorithmen und grammatikbasierten Systemen wird auf KI-gestützte Methoden eingegangen. Vorteile und Einschränkungen für Indie-Entwickler*innen werden analysiert, gefolgt von Fallstudien erfolgreicher Rogue-like-Spiele. Die Arbeit endet mit einem kurzen Ausblick auf die potentiellen zukünftigen Entwicklungen im Bereich der Procedural Generation.

1 Einleitung

Die Procedural Generation (PG) hat in der Spieleentwicklung eine Rolle eingenommen, welche nicht mehr wegzudenken ist. Dadurch, dass die Inhalte wie Levels, Charaktere oder Story-Elemente, dynamisch erstellt werden können, ermöglicht sie die Entwicklung von unterschiedlichen Spielwelten. Das ist insbesondere für Indie-Studios mit begrenzten Ressourcen von Vorteil, da diese oft von Geld-, Zeit- und Personal-Problemen betroffen sind. Beispiele wie Minecraft und No Man's Sky demonstrieren das Potenzial dieser Technologie. (Togelius et al., 2011; Shaker et al., 2016)

Es wird versucht, etablierte PG-Techniken zu analysieren und ihre Vor- und Nachteile darzustellen, sowie deren Umsetzung in der Praxis, insbesondere in Rogue-like-Spielen.

2. Vorteile der Procedural Generation für Indie-Spiele

Der wahrscheinlich größte Vorteil der Procedural Generation liegt in ihrer Effizienz. Entwickler*innen können Inhalte skalierbar und ressourcenschonend generieren, was besonders für Studios mit begrenzten finanziellen und personellen Mitteln entscheidend ist, was bei Indie-Studios meist zutreffend ist. Beispielsweise ermöglichen prozedural generierte Welten wie in „Spelunky", „Dead Cells" oder „Hades" nicht nur Kosteneinsparungen, sondern erhöhen auch die Wiederspielbarkeit von Spielen (Motion Twin, 2018; Supergiant Games, 2020; Adams, 2014). Level die zufällig erstellt wurden, bieten Spieler*innen eine größere Vielfalt und fördern die Motivation.

3. Herausforderungen und Einschränkungen der Procedural Generation

3.1 Kontrollverlust

Eine relevante Herausforderung der Procedural Generation ist der Kontrollverlust über die erstellten Inhalte. Unvorhergesehene Szenarien oder unausgewogene Schwierigkeitsgrade können das Spielerlebnis beeinträchtigen. Die Hauptschwierigkeit liegt darin, einen ausgewogenen Kompromiss zwischen Zufälligkeit und Struktur zu finden, um eine abwechslungsreiche, aber konsistente Spielerfahrung zu gewährleisten. Zu stark zufällig generierte Inhalte können den Spielfluss stören, während zu viele Regeln die kreativen Möglichkeiten der Procedural Generation einschränken (Gervás, 2005).

3.2 Technische Komplexität

Die Entwicklung und Implementierung prozeduraler Systeme erfordert spezialisierte Kenntnisse und kann zeitaufwendig sein. Fehlerhafte Inhalte können das Gameplay negativ beeinflussen. (Summerville et al., 2018)

3.3 Monotonie

Wiederholende Muster stellen ein Risiko für die Spielerfahrung dar. Spiele wie No Man's Sky haben anfangs Kritik für repetitive Inhalte erfahren. (Adams, 2014)

4 Techniken der Procedural Generation

4.1 Pseudozufallsgenerierung

Die Pseudozufällige Generierung von Zahlen kommt aus der Informatik. Zufällige Zahlen spielen laut Adams (2014) eine essenzielle Rolle im Kontext von Spielen. Damit jede*r Spieler*in eine eigene Form des Spieles erfährt, werden zufällige Zahlenketten verwendet, um die nötige Varianz zu erzeugen. (Adams, 2014)

Pseudozufällige Algorithmen bilden die Grundlage vieler PG-Techniken. Durch einen Seed-Wert lassen sich reproduzierbare Welten generieren, was besonders in Rogue-like-Spielen Anwendung findet. So können zufällige Level erstellt werden, die dennoch einer festgelegten Struktur folgen. (Gervás, 2005)

4.2 Perlin Noise und fraktale Algorithmen

Perlin Noise, entwickelt in den 1980er Jahren, ist ein Standard für natürliche Strukturen wie Terrain oder Wettereffekte (Togelius et al., 2011). Dieser Algorithmus erstellt ein Muster, welches nicht vollkommen auf Zufall basiert, sondern nur pseudo-zufällig. Diese Muster werden auch glatte Muster genannt und hängen zusammen. (Perlin, 1985)

Fraktale Algorithmen ergänzen diese Methode, indem sie durch Selbstähnlichkeit komplexe Strukturen erzeugen, wie sie auch in Spielen mit prozedural generierten Landschaften zu finden sind (Dead Cells, Motion Twin, 2018). Selbstähnlichkeit bedeutet in diesem Kontext, dass Strukturen ähnliche Muster aufweisen.

4.3 Grammatikbasierte Generierung

Laut Adams (2014) basiert diese Art der Generierung auf der Idee, dass Inhalte mithilfe von formalen Regeln erstellt werden, vergleichbar mit der Grammatik, welche unsere Sprache strukturiert. Dadurch werden Regeln definiert, sodass einzelne Elemente miteinander kombiniert werden dürfen, um eine Struktur zu erzeugen. (Adams, 2014)

Grammatikbasierte Ansätze nutzen Regeln und Constraints, um interessante Levelstrukturen zu schaffen. Ein Beispiel dafür ist „Spelunky", das durch eine Kombination von Zufall und Struktur einzigartige und herausfordernde Level generiert (Adams, 2014).

4.4 KI und Machine Learning

KI-gestützte Ansätze wie Generative Adversarial Networks (GANs) oder neuronale Netze ermöglichen die dynamische Erstellung und Anpassung von Spielinhalten (Goodfellow et al., 2020).

GANs sind eine Technik des Machine Learning, die auf zwei neuronalen Netzen basiert. Es werden Daten generiert, welche den Trainingsdaten ähneln. Diese Technik wird oft für Texturen, Umgebungen oder Charaktere verwendet. (Goodfellow et al., 2020)

Diese Technologien bieten eine enorme Flexibilität, sind jedoch ressourcenintensiv und stellen insbesondere für Indie-Studios eine Herausforderung dar. Hierfür wird leistungsstarke Hardware benötigt. Es wird ebenfalls fundiertes Fachwissen in den Bereichen Machine Learning und Datenverarbeitung benötigt. Die Entwicklung und Implementierung KI-basierter Systeme ist meist mit hohen Vorlaufzeiten und Kosten verbunden, was für kleinere Studios weniger zugänglich ist (Summerville et al., 2018).

5. Procedural Generation in Rogue-like Spielen

In den letzten Jahren hat sich die Procedural Generation besonders in Rogue-like Spielen durchgesetzt. Diese Spiele nutzen Zufallsmechaniken, um Spieler*innen immer wieder vor neue spannende Herausforderungen zu stellen. Nach Doan (2019) bestehen die Kernmechaniken eines Rogue-like Spiels darin, dass Procedural Generation genutzt wird, dass jeder Durchlauf ein neues unvorhersehbares Level gibt. Noch dazu ist die Mechanik des „Permadeath“ eines der wichtigsten Features. Dadurch sind Spieler*innen emotional mehr an ihren Charakter gebunden und müssen selbst einschätzen, welche Risiken genommen werden. (Doan, 2019)

Im Folgenden werden einige der bekannteren Beispiele vorgestellt.

5.1 Dead Cells

Dead Cells kombiniert Elemente eines Rogue-like mit einem so genannten Metroidvania-Stil. Die prozedural generierten Level bieten bei jedem Spieldurchgang unterschiedliche Anordnungen, was den Spieler*innen neue Strategien und Herangehensweisen abverlangt. Jedes Mal, wenn ein*e Spieler*in stirbt, wird eine neue Welt erschaffen, was die Wiederspielbarkeit stark erhöht und gleichzeitig ein Gefühl der Belohnung und Fortschrittlichkeit vermittelt. (Motion Twin, 2018)

Spieler*innen können durch freischaltbare Fähigkeiten neue Bereiche in Levels erreichen. Der relativ hohe Schwierigkeitsgrad wird durch ein Belohnungssystem ausgeglichen, bei welchem Spieler*innen die Währung „Zellen“ sammeln können, um permanente Verbesserungen zu kaufen. Dadurch erreichen Spieler*innen, selbst nach Scheitern eines Durchlaufes, Fortschritt. (Motion Twin, 2018)

5.2 Hades

Hades von Supergiant Games nutzt prozedural generierte Level, wobei sich die Anordnung der Räume jedes mal ändert und die Gegner variieren. Dadurch haben Spieler*innen ein frischeres Spielerlebnis und können das Spiel länger genießen. Bei jedem Durchgang erhält ein*e Spieler*in neue Dialoge und Interaktionen mit den Charakteren, was die Verbindung zur Geschichte stärkt und die Motivation erhöht, die Welt erneut und immer wieder zu erkunden. (Supergiant Games, 2020)

Hier begegnen Spieler*innen mehreren Charakteren aus der griechischen Mythologie, mit welchen man interagieren kann. Jeder Spieldurchlauf bietet neue Dialoge, die die Story vorantreiben, wodurch sich das Spiel nicht nur durch den Schwierigkeitsgrad, sondern auch durch die Geschichte auszeichnet. (Supergiant Games, 2020)

6. Ausblick

KI-gestützte Ansätze werden in der PG eine entscheidende Rolle spielen. Machine-Learning-Algorithmen könnten künftig personalisierte Spielinhalte ermöglichen, die auf das Verhalten einzelner Spieler*innen abgestimmt sind. Diese Entwicklungen könnten auch die Grenzen von Spielen überwinden und in anderen kreativen Industrien Fuß fassen. (Shaker et al., 2016)

6.1 Automatisierte Qualitätsbewertung

Ein entscheidender Schritt für die Zukunft der Procedural Generation könnte in der Entwicklung automatisierter Systeme zur Qualitätsbewertung liegen. Entwickler*innen, die mit generativen Systemen arbeiten, stehen vor der Herausforderung, nicht nur einzelne Inhalte zu bewerten, sondern das gesamte Spektrum möglicher Inhalte zu verstehen, welche aus einem System hervorgehen können. In der Arbeit von Adams (2014) wird betont, dass Entwickler*innen eben nicht nur den Output von generierten Inhalten manuell begutachten sollten, sondern dass die automatische Analyse der gesamten generativen Umgebung notwendig ist, um die Qualität und Diversität der Inhalte sicherzustellen. (Adams, 2014)

Automatisierte Bewertungsmechanismen könnten helfen, Systemfehler oder Verzerrungen zu erkennen, bevor diese in das endgültige Spiel gelangen.

7. Fazit

Procedural Generation bietet Indie-Entwickler*innen eine gute Methode, um umfangreiche und einzigartige Spielwelten zu erschaffen, die die Spieler*innen im besten Fall faszinieren und herausfordern. Insbesondere im Bereich der Rogue-like-Spiele hat sich PG als entscheidendes Element für Wiederspielbarkeit und Spielerengagement etabliert. Trotz technischer und gestalterischer Herausforderungen kann die richtige Balance zwischen Zufall und Kontrolle ein qualitativ hochwertiges Spielerlebnis fördern. Zukünftige Entwicklungen, insbesondere im Bereich Machine Learning, könnten das Potenzial der PG für die Indie-Entwicklung weiter steigern und die Kreativität der Entwickler*innen fördern.

Quellen

Adams, E. (2014). *Fundamentals of game design* (3rd ed.). New Riders.

Doan, D. (2019). Gamedev protips: How to design a truly compelling roguelike game. *Medium*. https://medium.com/@doandaniel/gamedev-protips-how-to-design-a-truly-compelling-roguelike-game-d4e7e00dee4

Gervás, P. (2005). Story plot generation based on CBR.

Goodfellow, I., Bengio, Y., Courville, A., & Bach, F. (2020). *Deep learning*. MIT Press.

Motion Twin. (2018). *Dead Cells* [Video game]. Motion Twin.

Perlin, K. (1985). An image synthesizer. *ACM SIGGRAPH Computer Graphics*, 19(3), 287–296.

Shaker, N., Togelius, J., & Nelson, M. J. (2016). *Procedural content generation in games: A textbook and reference*. Springer.

Summerville, A., Snodgrass, S., Guzdial, M., Holmgård, C., Hoover, A. K., Isaksen, (2018). *Procedural Content Generation via Machine Learning (PCGML)*

Supergiant Games. (2020). *Hades* [Video game]. Supergiant Games.

Togelius, J., Kastbjerg, E., Schedl, D., & Yannakakis, G. N. (2011). What is procedural content generation? *Proceedings of the 2nd International Workshop on Procedural Content Generation in Games.*