Prozedurale graphenbasierte 2D-Level-Generierung

Matutat André

20. Juli 2019

Matutat André (1119367)
Prozedurale graphenbasierte 2D-Level-Generierung
20. Juli 2019

Erstprüfer/in: Prof. Dr.-Ing. Carsten Gips , FH Bielefeld
Zweitprüfer/in: Dipl.-Inform. Birgit Christina George, FH Bielefeld

Zusammenfassung

Dies ist die Zusammenfassung auf Deutsch.

Abstract

This is an english abstract.

Inhaltsverzeichnis

1. Einleitung

Ein essenzieller Bestandteil eines jeden Videospiels sind seine Level. Level gehören zum Kernpunkt der Interaktion, sie sind die Spielwiese, auf der sich der Spieler mithilfe der Spielmechaniken austobt. Daher sind die Konzeptionierung und Gestaltung guter Level ein Zeit- und damit kostenintensives Unterfangen. Da Videospiele immer umfangreicher werden, müssen auch mehr Level produziert werden. Eine Möglichkeit, um kostengünstig eine Vielzahl an unterschiedlichen Level zu produzieren ist die Verwendung von Level-Generatoren, welche eigenständig unter Betrachtung verschiedener vorher definierter Kriterien, Level erzeugen können. Da jedes Videospiel sich von anderen unterscheidet und daher eigenen Anforderungen an seine Level stellt, müssen Level-Generatoren für jedes Projekt angepasst werden, um den Anforderungen des Spiels gerecht zu werden. Die Entwicklung eines solchen Generators kann je nach Anforderungen anspruchsvoller und teurer sein als die manuelle Gestaltung von Level. Ein gut entworfener Algorithmus ist jedoch in der Lage nahezu unendlich viele unterschiedliche Level zu generieren und so den Spielspaß aufrechtzuerhalten, vor allem das Genre der Rogue Like macht sich dies zu nutzen.

Im Rahmen des Moduls Programmiermethoden des Studienganges Informatik der FH-Bielefeld sollen die Studenten ihre Fähigkeiten mit der Programmiersprache Java und verschiedenen Konzepten und Tools der Softwareentwicklung verbessern. Hierzu findet neben den theoretischen Vorlesungsteil ein praktischer Programmierteil statt. In diesem praktischen Anteil entwickeln die Studenten ein eigenes 2D-Rollenspiel, das PM-Dungeon. [1] Hierfür bekommen sie das PM-Dungeon-Framework zur Verfügung gestellt.[2] Das Framework entlastet die Studenten, indem es die für das Lernziel des Moduls irrelevanten Inhalte übernimmt. Dazu zählt nebst der grafischen Darstellung auch das Laden der Level. In der Version *1.1.5.1* unterstützt das Framework nur Level welche durch den **Edgar-DotNet** Generator in der veralteten Version *1.0.6* erzeugt wurden.[3] Da dieser Generator nicht auf die spezifischen Anforderungen des PM-Dungeons optimiert wurde, beschäftigt sich diese Arbeit mit der Entwicklung eines neuen Generators.

Ziel dieser Arbeit ist es, bekannte und bereits etablierte Verfahren zur Generierung von 2D-Level zu analysieren und die jeweils besten Elemente der verschiedenen Algorithmen in einen neuen Graphen basierten Generator zu kombinieren. Die erzeugten Level sollen gängigen Kriterien für gutes Leveldesign erfüllen, daher werden im Rahmen dieser Arbeit solche Kriterien erläutert und ausgearbeitet. Der implementierte Algorithmus soll dann in das PM-Dungeon-Framework integriert werden und den Studenten verschiedene Möglichkeiten bieten, mit diesem Level zu arbeiten und ihre eigenen Spielelemente wie Items oder Monster mit dem Level zu verbinden. Die Studenten sollen auch in der Lage sein, den Algorithmus so zu konfigurieren, dass sie Level

nach eigenem Belieben gestalten können. Das bedeutet, die Studenten müssen in der Lage sein, den Aufbau der Level dynamisch abfragen zu können um Beispielhaft den kritischen Pfad, also den Pfad vom Spieler bis zum Ziel bestimmen zu können, um entsprechend Monster zu platzieren. Der Generator soll möglichst spaßige und abwechslungsreiche Level erzeugen.

Die Implementation der Spiellogik ist nicht Bestandteil dieser Arbeit. Ebenso konzentriert sich diese Arbeit nicht darauf, einen Laufzeit oder Speicherplatz optimierten Algorithmus zu entwickeln. Diese Arbeit beschäftigt sich ausschließlich mit der Generierung von Level im zweidimensionalen Raum, 3D-Level und Spiele werden aufgrund ihrer stark abweichenden Anforderungen zu 2D-Level nicht betrachtet. Grafiken und Soundeffekte werden nicht generiert, jedoch soll der Generator in der Lage sein zur Verfügung gestellte Texturen zu verwenden.

todo aufbau der arbeit

2. Stand der Technik/Forschung, vergleichbare Arbeiten

2.1. PM-Dungeon

- Was is Rogue
- Was sind Rogue Likes?
- Was ist das PM-Dungeon
- Welches Ziel soll das PM-Dungeon erfüllen
- Was ist das PM-Dungeon-Framework?
- Welche Probleme hat das Framework aktuell noch bezogen auf Level

2.2. Prozedurale Generierung

- Begrifferläuterung
- Anwendungsfälle und Beispiele
- Anwendung zur Level generierung
- Beispiele welche NICHT im Kapitel Analyse verwendet werden, daher auch kurz und knall (z.B Random Walk)
- Granzen und Probleme
- Vor und Nachteile

2.3. Bewertungskriterien für gute Level

- Aufstellen von Bewertungskriterien
- Fokusiert auf prozedurale Anwendungsfälle
- Fokusiert auf Rogue-Likes
- Fokusiert auf 2D-Spiele
- Kürzer als in der BA

2.4. Graphen zur darstellung von Level

- Wie verwendet man Graphen zur darstellung von Level?
- Warum ist das gut?
- Welche Vortiele bietet das?
- Welche Nachteile und Grenzen gibt es?
- Warum mach ich das in dieser arbeit so?

3. Analyse

*Für jeden Algo*rithmus

- Wie funktioniert der Algorithmus?
  - evtl. Technologie erklären (hier oder wo anders?)
- Zeigen von Beispielen
- Bewerten das Algorithmus anhand der Kriterien
  - Was macht der Algo gut?
  - Was macht der Algo schlecht?
  - Welche grenzen gibt es?

3.1. Vergleich der Algorithmen

- als Matrix?

4. Eigene Ideen, Konzepte, Methoden

4.1. Zielsetzung und Anforderungen an das Projekt

- Was genau muss der Algo können?
- Was nicht?
- Welche Anforderungen muss der Algorithmus und das Framework erfüllen?

4.2. Bewertungsverfahren

- Kriterien aus Kapitel 2
  - Manuelle Analyse
- Umfrage bei Studenten (Zeitplanung?)

4.3. Bausteine

- Auflisten der der Bausteine aus Kapitel 3
- Begründen warum diese verwendet werden

4.4. Abgrenzung zu den anderen Algorithmen

- Warum ist dieser Algo besser für das PM-Dungeon als die anderen?
- Welche Grenzen wird dieser Algo haben im vergleich zu den anderen?

4.5. Darstellung des Konzeptes

- WIe werden die Bausteine kombiniert?
- Welche Probleme treten dabei auf?
- Welche anpassungen müssen daher gemacht werden?
- Zusätzliche Features die NICHT aus den Algorithmen stammen (vermutlich die API für die Studenten)

5. Realisierung

5.1. Umsetzten der Bausteine

- Wie?
- Welche Probleme?

5.2. Umsetzten der Verbindungen

- Wie?
- Welche Probleme?

5.3. Umsetzten der API

- Wie?
- Welche Probleme?

6. Evaluierung

- Zeigen von Ergebnissen
- Auswerten der Ergebnisse

7. Zusammenfassung

7.1. Fazit

- Wurde das Ziel erreicht?
  - Vergleich mit der ursprünglichen Zielsetzung
  - Was kann ich jetzt genau mit dem Algo machen?
- Sachen die gut gelaufen sind
  - Warum sind die gut?
- Sachen die schlecht gelaufen sind
  - Warum sind die schlecht gelaufen?
- Welchen Beitrag hat diese Arbeit für das “Forschungsfeld” prozedurale Generierung geboten?

7.2. Ausblick

- Wie kann man den Algo erweitern?
- Wie geht die prozedurale Generierung in der Praxis weiter?
  - Graphenbasiert
  - Nicht graphenbaiser

Literatur

1. 2018. *Studiengangsprüfungsordnung für den BachelorstudiengangInformatikan der Fachhochschule Bielefeld(University of Applied Sciences)vom 20. September 2018 in der Fassung der Änderungvom 05. Juni 2019*.

2. André Matutat. 2021. PM-Dungeon on Git. Abgerufen von https://github.com/PM-Dungeon/pmdungeon.

3. Ondrej Nepozitek. 2017. Edgar-DotNet on Git. Abgerufen von https://github.com/OndrejNepozitek/Edgar-DotNet.

A. Appendix 1: Some extra stuff

A.1. Level

- Graphen und draus erzeugte Level

A.2. Code

- Vermutlich ehr nicht vorhanden

A.3. Umfrage

- Umfrageergebnisse in voller länge

B.

Eidesstattliche Erklärung

Ich erkläre hiermit an Eides Statt, dass ich die vorliegende Arbeit selbstständig angefertigt habe. Die aus fremden Quellen direkt oder indirekt übernommenen Gedanken sind als entsprechend kenntlich gemacht.

Ich habe die Arbeit bisher weder in gleicher noch in ähnlicher Form einer anderen Prüfungsbehörde vorgelegt und auch noch nicht veröffentlicht. Das PDF-Exemplar stimmt mit den eingereichten Exemplaren überein.

Minden, den 20. Juli 2019