Dokumentation: Videospiel  
Von Grigorii Kostromin und David Hofmann

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Datum | Tätigkeit | Ergebnisse/Notizen | Erfolge | Probleme |
| 14.01.2025 | Git/GitHub Einarbeitung | * Erstellen einer Repository * Benutzen von git-Befehlen * Auf die Repository pushen, pullen und Dateien bearbeiten | Wird uns ermöglichen einfacher und auch zeitgleich am Projekt zu arbeiten und auf verschiedenen Geräten die aktuellste Version des Projekts zu haben. So können wir schneller arbeiten, ohne uns sorgen darum zu machen es irgendwo extern zu speichern. | Zunächst Verständnisprobleme und die erstmalige Einrichtung bei Github. Jedoch klärten sich die Probleme. |
| 15.01.2025 | Installation & Wahl einer Python-Bibliothek | * Python, tcod installieren * Erste Python Datei erstellen * Vertraut machen mit Visual Studio Code | Verständnis über die nötigen Installationen erlangen, damit wir wissen, was wir genau im Projekt verwenden werden können.  Auch nach etwas Recherche fanden wir heraus, dass tcod häufig bei Spielen ähnlicher Art verwendet wird, weswegen wir uns für diese Bibliothek entschieden. Nun haben wir tcod als Basis. | Keine |
| 18.01.2025 | Spieler erzeugen | * Erstmalig den Spieler in der „Welt“ erscheinen lassen * Bewegung des Spielers implementieren | Auffrischung von den Grundlagen Pythons und erstmalige Berührungspunkte mit tcod. Nun haben wir unser erstes Element in der Welt und wissen, wie man mit tcod Tasteninputs verarbeitet. | Keine |
| 21.01.2025 | Git/GitHub Wiederholung in der Schule | * Klonen der Repository auf dem Schulrechner * Benutzen von git auf dem Schulrechner | Git erstmalig auf den Schulrechnern ausprobieren und schauen, wie gut wir unser Projekt auf die Computer holen können, sodass wir auch effektiv im Unterricht daran arbeiten können. | Anfängliche Probleme beim Login, da auf Linux die Anmeldung über die Befehlszeile anders geregelt wird.  -> Lösung: Einen Login Code erstellen, mit dem wir uns schnell auch auf den Schulrechnern anmelden können, sodass wir das Projekt in der Cloud speichern und später wieder auf die anderen Geräte übertragen können |
| 22.01.2025 | Erzeugen eines anderen Objektes (NPC) | * Erstmalige Einführung von Entitäten als Klasse * Jegliche Objekte können später aus der Klasse erzeugt werden | Die „Entity“ Klasse bietet für uns einen wichtigen Grundbaustein, da wir später auch planen Gegenstände, wie Tränke, Zauber, aber auch Gegner ins Spiel einzubauen, die alle denselben Regeln „gehorchen“ müssen, weswegen wir das über eine gemeinsame Klasse abwickeln wollen | Keine |
| 28.01.2025  +  29.01.2025  +  31.01.2025 | Erzeugen eines Spielfeldes | * Spieler ist in einem geschlossenen Raum * Tiles als eigene Klasse * Definition von Wand und Boden * Renderfunktion, um die Räume sehen zu können | Hier haben wir nun beispielsweise die erste Regel, der alle Entitäten gehorchen müssen: Sie dürfen nicht in Wände rein. So haben wir nun die Unterteilung der Welt in die „Kacheln“ und wir können beliebig über die Tile-Klasse Tiles in der Welt platzieren, die für beispielsweise den Spieler nicht begehbar sind | Probleme mit Renderfunktion |
| 02.02.2025 | Spielfeld Experimente | * Manuell Wände erzeugen * Manuell Räume erzeugen * Manuell Gänge erzeugen | Das Vertraut-Machen mit unserer Tile Klasse und wie wir am einfachsten Räume/Wände erzeugen können.  Entschluss: Bestenfalls ist jede Kachel in der Welt nicht begehbar und nicht sichtbar für den Spieler -> dann setzen wir die von uns definierte Variable, die die Begehbarkeit kontrolliert auf False, damit der Spieler sich auf den ausgesuchten Feldern bewegen kann. -> Bisher aber nur „hardcode“ möglich. Später möchten wir eine zufällige Generation einbauen | Keine |
| 06.02.2025 | Überlegung zur Raumgeneration | * Plan, wie wir die Räume generieren wollen | Entwerfen eines Konzeptes, wie die Räume automatisch generiert werden sollen. Wir wollen ein Rechteck mithilfe zufällig generierten x- und y-Koordinaten einer oberen linken und dann einer unteren rechten Ecke erzeugen, wonach wir dann alle in diesem Bereich enthaltenen Kacheln in eine Liste packen, wo wir dann die Variable der Begehbarkeit True setzen. Ähnlich läuft dann die Generation der Tunnel, die die Räume verbinden, jedoch werden hier dann keine Koordinaten zufällig generiert, sondern die Mitten der Kanten der Räume genommen und dann schnellstmöglich mit dem danach generierten Raum verbunden. | Keine Implementation |
| 08.02.2025 | Besprechung des Konzeptes | * Verbesserungen an Konzept | Weitere Ausarbeitung des oben beschriebenen Konzepts (steht bereits niedergeschrieben) | Keine |
| 09.02.2025  +  10.02.2025 | Implementierung des  Konzeptes | * Konzept implementiert | Hier haben wir das Konzept mit den zufällig gewählten Rechtecken und den verbindenden Tunneln ins Spiel implementiert, sodass nun der Spieler eine Welt hat, in der er sich fortbewegen kann | Anfangs nicht funktioniert, jedoch Fehler behoben |
| 12.02.2025 | Zufälliges Platzieren von Entitäten in der Welt | * Das zufällige Platzieren von Entitäten in der Welt in kooperiert in die vorhererklärte Welterstellung | Jetzt wird der vorhererstellte NPC (es ist eigentlich nur eine gelbe Version des Spielers, die sich nicht bewegen kann), zufällig in der Welt erzeugt, wo er nun einfach dasteht. Das ist ein Erfolg, da wir jetzt dasselbe mit dem Spieler machen können und später auch mit den Gegenständen und Gegnern. | Keine |
| 13.02.2025 | Auskommentieren | * Die Weltgeneration und weitere vorher unkommentierte Aspekte sind jetzt kommentiert | So können wir uns an den Code, den wir vor einiger Zeit schrieben, besser erinnern und das ist auch gut für das Verständnis von Dritten, die sich den Code ansehen wollen. | Keine |
| 14.02.2025 | FOV (Field-Of-View) und Rendering der Welt | * Eingeschränkte Sicht des Spielers und „Erkundung“ möglich. | Eine andere der von uns geplanten Grundfunktionen ist jetzt im Spiel, wir können jetzt nur in einem limitierten Bereich sehen und müssen quasi die Karte erkunden und müssen uns überraschen lassen was vor uns liegt. | Keine |
| 17.02.2025- 18.02.2025 | Zufälliges Platzieren von Gegnern und erste Umrisse des Kampfsystems | * Es gibt jetzt die Klasse Gegner, sie werden in der Welt platziert und es kann erkannt werden, wenn der Spieler gegen sie läuft. | Der von uns zufällig in der Welt generierte NPC wird jetzt zu einem simplen Gegner umgeformt, gegen den der Spieler laufen kann und es vom System erkannt wird. Es gibt jetzt eine Klasse für die Gegner und es wird beim Bewegen des Spielers geprüft, ob die Position wo der Spieler sich hinbewegen möchte, schon von einem Gegner belegt ist. Wenn ja, dann bewegt er sich nicht dahin, aber fürs erste wird dann erstmal etwas geprinted. | Umwandlung des NPCs, der in der Mapgeneration-Methode erstellt wird in einen Gegner, der zu einer Klasse gehört und dann erst auf der Karte erstellt wird. |
| 20.02.2025-  02.03.2025 | Kampfsystem & „KI“ zum Bewegen der Gegner & auch die „Spielzüge“ | * Zusatzklasse für die Gegner & auch Spieler („Fighter“) * Erstellung der Variablen nötig für den Kampf * Abrechnung nach Attacken. * Das Fortbewegen der Gegner * Der Spieler und die Gegner wechseln sich jetzt in den Spielzügen ab | In der Klasse Fighter können wir abhängig von was für einer Entität das ist, die diese Klasse angehangen bekommen folgende Variablen definieren: max\_hp, hp, defense und auch power. Das wird uns beim Kampf die Abrechnung ermöglichen.  Es wird jetzt jedes Mal nach dem Bewegen des Spielers „der Zug gewechselt“ und die „move\_towards“ Funktion der Gegner wird abgerufen. Bei der wird die Distanz zu dem Spieler berechnet, indem wir die Distanz, auf der x- und die Distanz auf der y-Achse berechnen, dann wird der Satz des Pythagoras angewandt, also die Distanz x und die Distanz y werden quadriert, addiert und danach die Wurzel gezogen, dann haben wir die Distanz. Danach wird die Richtung bestimmt, wo der Spieler ist und danach einen Schritt dahin getätigt.  Außerdem wird, wie auch beim Spieler, geprüft ob sich der Spieler dort befindet, wenn ja findet jetzt die Verrechnung statt. Dabei werden von der Power die Armor abgezogen und dann das, was übrig ist von den hp abgezogen.  Außerdem wird geprüft, ob sich die Gegner in dem FOV des Spielers befinden, dann können sie sich nur bewegt  Außerdem wenn man keine Leben mehr hat, ist man tot, und man kann sich nicht mehr bewegen. Dasselbe mit Gegnern, dessen Sprites sich zu einem „%“ verändert.  Auch wird jetzt mithilfe von tcod die Position der Maus getracked, und der Name der Entität, über der sich die Maus befindet, wird unten links im Bildschirm dargestellt.  Außerdem gibt es unten links auch eine Lebensanzeige, die auch mithilfe von tcod erstellt wird. Es ist quasi eine Leiste einer bestimmten Länge erzeugt, dann die hp, die der Spieler im Moment hat, im Vergleich zu der max\_hp in Prozent umgewandelt und dann die Balken Länge anhand dessen angepasst. Die standartlänge bleibt in einem blassen Rot da. | Probleme allein bei der Entwicklung der Idee, wie man die Bewegung der Gegner regeln möchte -> Belesung im Internet, dabei auf das Konzept mit dem Pythagoras gestoßen.  Weiteres Problem: Der Gegner hat Probleme um eine Ecke zu kommen. |
| 05.03.2025 | Behebung des Eckenproblems | * Verwenden von „move\_astar“ | Eine ausführliche Dokumentation von move\_astar steht bei uns als Kommentare im Code. | Keine |
| 09.03.2025-  10.03.2025 | Interface Elemente | * Konsole im Unteren Teil des Bildschirms * Nachrichtenausgabe | Mithilfe der tcod Bibliothek kann man ein Fenster erzeugen, in welchem bestimmte Strings ausgegeben werden können. Das ist gut, da der Spieler so einen guten Überblick über das Geschehen in der Welt behalten kann und es ist der beste Weg wie die Entwickler dem Spieler Nachrichten mitteilen können. So haben wir zu den verschiedensten Geschehnissen, wie der Schadensberechnung, oder wenn ein Gegner stirbt. | Keine, man musste sich nur etwas mehr in tcod einfuchsen |
| 11.03.25 –  12.03.2025 | Weiteres Interface & Items, sowie Inventar | * Eine Inventaranzeige * Heiltrank als erster funktionsfähiger Gegenstand * Items werden zufällig platziert | Zunächst gibt es nun eine Liste für Gegenstände, die auch zu der Entitätsklasse gehört. Jetzt haben wir Wahrscheinlichkeiten und Maximalanzahlen für Gegenstände und von Gegenständen in einem Raum, wo wir erstmal ein einfaches Ausrufezeichen als Platzhalter hinprinten.  Danach haben wir jetzt einen neuen möglichen Tasteninput, g, mit dem wir den Gegenstand „aufheben“ können, wenn die Koordinaten des Spielers und des Gegenstandes dieselben sind. Zunächst verschwindet der Gegenstand aber einfach nach dem Drücken von g.  Auch machen wir eine Inventarklasse, wo eine Maximalanzahl an Items, die der Spieler tragen kann, festgelegt wird und auch eine Funktion, falls er zu viele trägt.  Mithilfe von tcod erstellen wir auch ein Fenster, wo die Gegenstände aufgezählt werden, die wir tragen. Das können wir mit „s“ sichtbar machen und dabei können wir uns auch nicht bewegen.  Hier verwenden wir auch args und kwargs, wo ein Tasteninput (Bsp. a, oder b) dann in einen Index übersetzt wird, womit wir quasi entscheiden welchen Gegenstand wir jetzt verwenden wollen.  Jetzt kann man auch den Trank verwenden und eine festgelegte Zahl wird den derzeitigen hp des Spielers hinzugefügt, dabei darf natürlich nicht die max\_hp überschritten werden.  Dieses Menü des Inventars haben wir quasi einfach kopiert, nur mit der drop Funktion, sodass wir mit „l“ ein ähnliches Menü öffnen können, mit dem wir Gegenstände erneut wieder fallen lassen können. | Keine |
| 13.03.2025 | Neue Gegenstände | * Feuerballzauber * Selbstzielender Blitzzauber | Wir haben jetzt auch Kampfgegenstände, die das Kämpfen interessanter machen als das bloße draufgehaue von zuvor. Es werden neue Gegenstände auf der Karte platziert.  Beim Blitzzauber, ähnlich wie dem pathfinding der Gegner, wird der nächste Gegner gefunden, dem dann ein massiver Schadenssatz hinzugefügt wird.  Mit dem Feuerball geht man in einen extra definierten Zug, wo man sich erneut nicht bewegen kann, man aber mithilfe der Maus eine Zielkachel auswählen kann, wo dann jeder (auch der Spieler) Schaden erhält. | Keine |
| 16.03.2025 | Speichern und laden | * Man kann speichern * Laden * Und man hat ein Startmenü, wo wir auswählen können was wir tun wollen. | Wenn wir ESC drücken, also ins Menü gehen, was wir erneut mithilfe von der tcod Bibliothek erstellt haben, wird die „save“ Funktion ausgeführt, wo SEHR VIELE Variablen gespeichert werden. Dazu gehören also alle x- und y-Koordinaten der Gegenstände, Gegner und des Spielers, sowie den derzeitigen Leben und auch der Liste an begehbaren und gesichteten Kacheln der Welt, wie auch der Liste von Gegenständen, die der Spieler trägt.  Die wird jedes Mal überschrieben, wenn man das tut, oder stirbt.  Beim Laden werden die gespeicherten Variablen „eingesetzt“.  Das ist gut fürs Spiel, da wenn man mal am PC was anderes machen muss, man ganz problemlos das Spiel schließen kann und vielleicht in einem Durchlauf wo man besonders weit gekommen ist, weiterspielen kann. | Probleme bei der Dateibenennung und dem Auslesen der Daten. Die Datei hieß „savegame.dat“ und beim Spiel laden wurde nach „savegame.dat.dat“ gesucht, weil wir einen Fehler bei dem Formulieren der Dateinamen im Code gemacht haben. |
| 17.03.2025 | Mehrere Etagen | * Man kann nun Treppen finden und weiter in das Dungeon hinabsteigen. | Es wäre langweilig hätte man bloß eine einzige Spielfläche, wo man einmal durchrennt, und dann hat man das Spiel durchgespielt. Nun haben eine unendliche Welt, wo man immer tiefer hinab gehen kann und immer neue Welten generiert werden. So kann man jetzt probieren möglichst weit im Spiel zu kommen. | Keine |
| 18.03.2025 | Gegner Generation Veränderung | * Wahrscheinlich, dass stärkere Gegner tiefer auftauchen * Wenn es einen starken Gegner oder Gegenstand gibt, dann ist es unwahrscheinlicher, dass mehr starke Gegenstände/Gegner im selben Raum auftauchen | Das ist gut für den Spielspaß, weil man so mehr Variation hat.  Außerdem haben wir jetzt das Spielprinzip geändert, von dem eher sehr taktischen, rennen und probieren den Gegner zu treffen, ohne selbst erwischt zu werden zum eher taktischen Umgehen mit seinen Ressourcen, öfter auf Fernkampf setzen und sich eher zu entscheiden, wann man in einen Kampf geht. Jetzt haben alle mehr Leben und man braucht mehr Schläge, um einen Gegner zu bezwingen. | Keine |
|  |  |  |  |  |