Tetris mit PyGame – Projektdokumentation

Orkun Öztürk

HTW Berlin – Informatik 3, Sommersemester 2025

27. Juli 2025

Inhaltsverzeichnis

1 Einführung

1.1 Motivation

Das Projekt zum Entwickeln eines Tetris-Games entstand aus dem Wunsch, vielleicht eines Tages in der Gaming-Branche zu arbeiten. Durch eine Exkursion mit dem Englischkurs von Thomas O'Mera, bei der ein Game-Development-Office besucht wurde, wurde uns Studierenden oft ans Herz gelegt, klein anzufangen und zu schauen, was man aus kleinen Projekten machen kann.

1.2 Was wurde entwickelt?

Ein eigenes Tetris-Projekt mit PyGame. Es gibt ein Menü mit Schwierigkeitsauswahl, ein Punkte-Tracking und natürlich das eigentliche Spiel mit allem, was dazugehört: Steine bewegen, rotieren, stapeln – und wenn's zu voll wird, kommt ein Game Over.

1.3 Warum Tetris?

Tetris erschien wie ein guter Einstieg. Es ist bekannt, bringt genug Logik mit, um was draus zu lernen, aber ist gleichzeitig nicht komplett überfordernd. Außerdem macht es Spaß, und das war auch ein Ziel: etwas bauen, was man am Ende wirklich benutzen und spielen kann.

2 Anforderungen

2.1 Personas

Persona 1: Lara, 17, Schülerin

Lara geht in die 11. Klasse auf einem Gymnasium in Hannover. Sie liebt Retro-Ästhetik, spielt gerne zwischendurch mal ein paar Runden Casual Games und entdeckt gerade Programmieren mit Python. Sie findet es cool, wenn Spiele simpel und direkt losgehen – ohne Account-Zwang oder komplizierte Steuerung. Tetris kennt sie aus Memes und von Ihrem großen Bruder.
Ziel beim Spiel: Einfach mal abschalten nach der Schule – und dabei das Gefühl haben, dass sie mit jedem Versuch besser wird.
Technischer Hintergrund: Erste Python-Erfahrung im Schulunterricht, interessiert sich für Game Dev.

Persona 2: Murat, 42, Montagenarbeiter

Mit seiner Frau und zwei Kindern lebt er ein bescheidenes Leben in Köln. Er arbeitet im Schichtdienst, mag aber Technik und hatte früher mal Spaß an Tetris auf seinem altem Nokia. Abends sitzt er manchmal am Familien-PC oder Tablet, schaut YouTube oder klickt sich durch einfache Spiele. Er spricht gut Deutsch, aber lieber einfach gehalten – keine langen Erklärungen.
Ziel beim Spiel: Ein bisschen Nostalgie genießen, Stress abbauen, vielleicht zeigen, dass man noch gut und schnell im Kopf ist und seinen alten Highscore besiegen kann.
Technischer Hintergrund: Keine Programmiererfahrung, nutzt aber regelmäßig Handy, PC und einfache Spiele.

Persona 3: Sanjivan, 24, Medieninformatik-Student

Er studiert Medieninformatik im 5. Semester und ist eigentlich ständig von Projekten umgeben. Gerade sucht er nach Inspiration für seine eigene Spiele-Idee und spielt gerne kleinere, selbstentwickelte Games von Kommiliton:innen durch. Er achtet auf Usability, Look & Feel und liebt es, durch Code zu scrollen und Fehler zu finden, die man zusammen ausbessern kann.
Ziel beim Spiel: Spaß haben, aber gleichzeitig analysieren, wie die Logik aufgebaut ist – und wie man es vielleicht noch optimieren könnte.
Technischer Hintergrund: Gute Python-Kenntnisse, arbeitet mit PyGame, Unity und hat schon kleinere Games selbst entwickelt.

2.2 User Stories

User Story für Lara

Als Schülerin mit Interesse an Retro-Games
kann ich Tetris direkt ohne Anmeldung und komplizierte Menüs spielen,
sodass ich schnell loslegen und mich nach der Schule entspannen kann.

User Story für Murat

Als berufstätiger Familienvater mit wenig Zeit und Technik-Erfahrung
kann ich Tetris einfach am PC starten, ohne viel Erklärung oder Setup,
sodass ich abends entspannen, in Erinnerungen schwelgen und versuchen kann, meinen alten Highscore zu knacken.

User Story für Sanjivan

Als Medieninformatik-Student mit Interesse an Game-Design und Usability
kann ich das Spiel nicht nur spielen, sondern auch den Code anschauen und analysieren,
sodass ich daraus Ideen für eigene Projekte ziehen und eventuell gemeinsam mit anderen Verbesserungen umsetzen kann.

2.3 Use Cases

Use Case 1: Lara – Tetris einfach starten und spielen

Akteur: Lara, Schülerin
Ziel: Schnell und unkompliziert Tetris spielen
Beschreibung: Lara öffnet das Spiel auf ihrem Laptop, klickt auf „Easy“ im Menü und startet sofort eine Runde. Sie möchte keine Registrierung, keine Tutorials, einfach direkt loslegen. Während des Spielens sieht sie ihre Punkte und freut sich, wenn sie besser wird. Wenn sie verliert, kann sie direkt neu starten.
Voraussetzungen: Spiel ist installiert und funktioniert ohne Internet.
Nachbedingung: Lara kann das Spiel beliebig oft neu starten und verbessern.

Use Case 2: Murat – Feierabend, entspannen, Highscore jagen

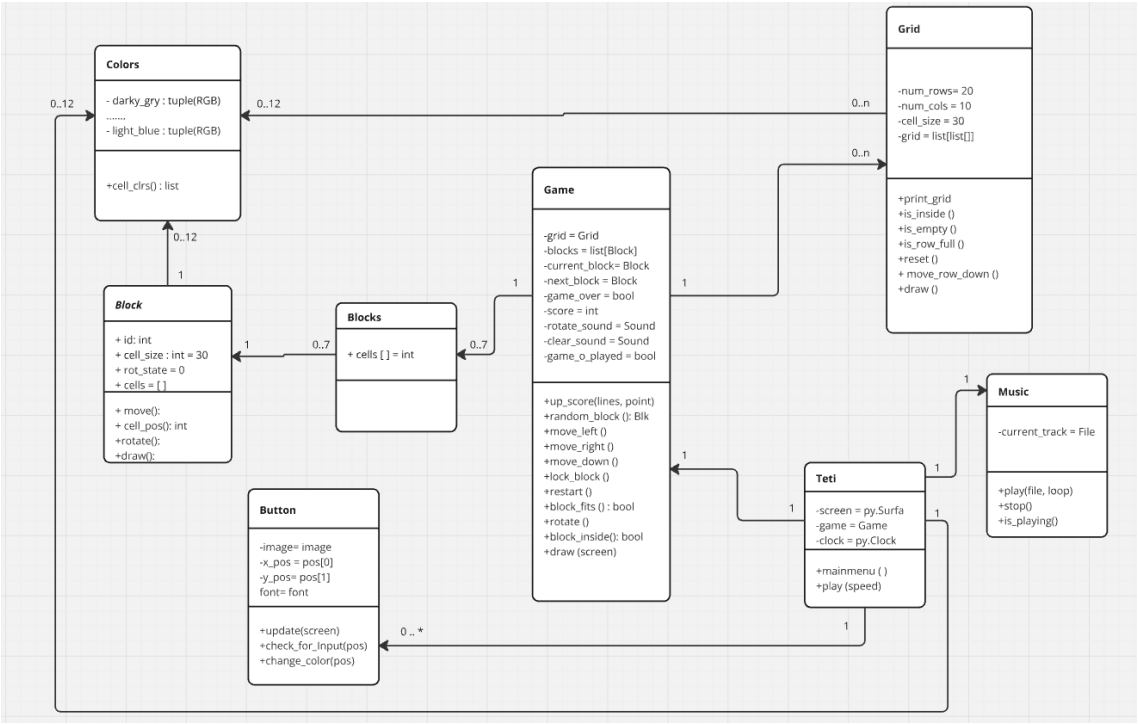
Akteur: Murat, Familienvater
Ziel: Kurze, entspannte Runde Tetris nach der Arbeit
Beschreibung: Nach der Spätschicht setzt sich Murat an den Familien-PC. Er klickt auf das Tetris-Spiel, wählt „Hard“, weil er sich der Herausforderung stellen will, und versucht, seinen alten Highscore zu schlagen. Die Steuerung ist leicht verständlich, es gibt keine unnötigen Texte. Wenn er verliert, bekommt er einen Game-Over-Screen mit Punktestand.
Voraussetzungen: Das Spiel startet schnell und läuft flüssig auf älteren Geräten.
Nachbedingung: Murat kann nach der Runde seinen Score vergleichen und neu beginnen.

Use Case 3: Sanjivan – Code verstehen und Spiel analysieren

Akteur: Sanjivan, Medieninformatik-Student
Ziel: Das Spiel sowohl spielen als auch im Quellcode nachvollziehen
Beschreibung: Sanjivan startet das Spiel, testet es in verschiedenen Schwierigkeitsstufen und achtet dabei gezielt auf Spielverhalten, UI-Details und eventuelle Bugs. Danach schaut er sich den Code an (z.B. in PyCharm), prüft die Logik und überlegt, wie man die Struktur verbessern oder erweitern könnte – zum Beispiel mit einem Highscore-System oder Level-Up-Animationen.
Voraussetzungen: Der Quellcode ist sauber strukturiert und kommentiert.
Nachbedingung: Sanjivan kann eigene Verbesserungen vorschlagen oder implementieren.

3 Design und Implementierung

3.1 UML-Diagramme



3.2 Technologien (PyGame, Python)

Für das Projekt wurde hauptsächlich mit Python gearbeitet – eine Sprache, die bisschen angeschnitten wurde aus anderen Modulen und die sich gerade für Spieleprojekte super eignet, weil man schnell sichtbare Ergebnisse bekommt.
Das Framework PyGame war dabei unsere erste Wahl. Es bietet viele fertige Funktionen für Dinge wie Grafiken zeichnen, Events abfragen oder Musik abspielen. Gerade für ein Spiel wie Tetris, das ständig auf Tastendrücke reagieren und visuelle Updates machen muss, hat PyGame ziemlich gut gepasst. Auch Soundeffekte und Hintergrundmusik ließen sich relativ einfach einbinden – wobei es da am Anfang auch kleinere Stolpersteine gab (z.B. das Stoppen der Musik beim Game Over war trickier als gedacht).
Zu Beginn gab es bisschen Respekt vor der ganzen Strukturierung, aber durch PyGame lernt man recht schnell, wie man Klassen wie **Game**, **Grid** oder **Block** sinnvoll trennt und miteinander kommunizieren lässt. Mit jedem neuen Feature (Rotation, Punkte zählen, nächste Form anzeigen) wurde das Ganze immer klarer.
Die Entwicklung war lokal auf verschiedenen Rechnern mit **pygame==2.5.2** installiert, und ein bisschen Herumprobieren mit Pfaden und Sounds hat manchmal nötig gewesen – aber insgesamt lief es ziemlich gut.

3.3 Wichtige Klassen und Logik

Im Zentrum des Spiels steht die Klasse **Game**. Sie hält eigentlich alles zusammen: das Spielfeld, den aktuellen Block, den nächsten Block und den Score. Von hier aus werden alle Bewegungen gesteuert nach links, rechts, runter, rotieren und auch geprüft, ob das Spiel vorbei ist.
Das **Grid** ist unser Spielfeld, bestehend aus einem 2D-Array mit 20 Zeilen und 10 Spalten. Hier wird jeder Block „eingetragen“, sobald er unten angekommen ist. Außerdem kümmert sich die Klasse darum, komplette Reihen zu erkennen und zu löschen – ein zentraler Bestandteil von Tetris.
Die Blöcke selbst sind als eigene Klassen wie `IBlock`, `TBlock` oder `ZBlock` umgesetzt. Sie erben alle von einer gemeinsamen **Block Klasse**, die die Bewegung und Rotation organisiert. Die Positionen der einzelnen Teilsteine sind als **Position**-Objekte gespeichert und verändern sich, wenn man den Block bewegt oder dreht. Jede Rotation ist dabei als vordefinierte Konfiguration abgespeichert.
Außerdem gibt's eine **Colors**-Klasse, die nur für die Farben zuständig ist – einfach damit wir den Überblick behalten und alle Farben an einer Stelle definieren konnten.
Für das Menü wurde eine einfache **Button**-Klasse gebaut. Die kümmert sich darum, ob man gerade mit der Maus drüber fährt oder drauf klickt. Beim Klick wird dann das Spiel gestartet oder beendet.
Die Spiellogik basiert auf einem Timer, der alle paar Millisekunden ein Event feuert – dadurch fallen die Blöcke automatisch nach unten, ohne dass man irgendwas drücken muss. Und mit jeder Bewegung oder beim Drehen wird auch geprüft, ob der Block noch im Feld ist und ob er auf andere Blöcke trifft.
Was anfangs ein Problem war: die Rotation so zu bauen, dass Blöcke nicht aus dem Spielfeld „rausrutschen“ oder in andere reingehen, aber das wurde dann nach einigen Tests stabil.

4 Ergebnisse

4.1 Screenshots der Anwendung

Abbildung 1: Hauptmenü mit Schwierigkeitsauswahl

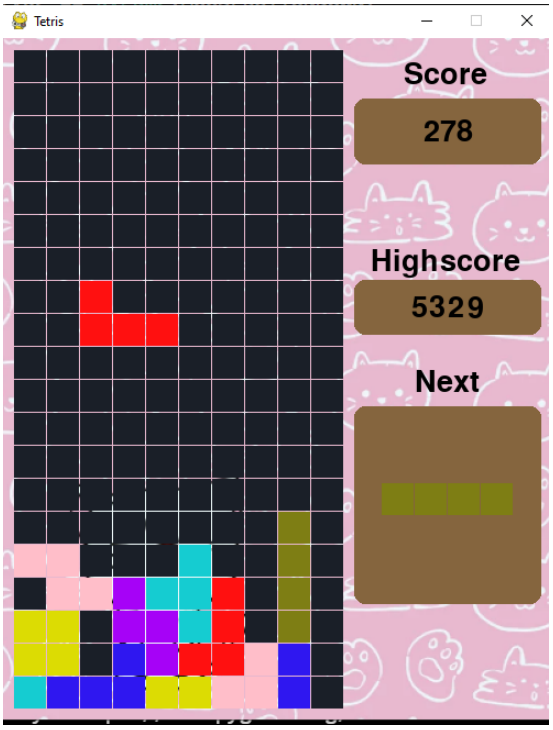


Abbildung 2: Spielbildschirm im einfachen Modus

4.2 Besonderheiten / Herausforderungen

Die größte Herausforderung im Projekt war die Implementierung einer stabilen Kollisionserkennung insbesondere in Kombination mit der Rotation der Blöcke. Es gab mehrere Situationen, in denen ein Block direkt neben einem anderen rotiert werden sollte, was zu Problemen geführt hat. In manchen Fällen kam es dadurch sogar zum Programmabsturz, da die Rotation nicht korrekt rückgängig gemacht wurde oder die Blockposition anschließend außerhalb des Spielfelds lag.
Zu den Besonderheiten des Projekts zählt auf jeden Fall die Integration von Musik und einem Menü-Design. Je nach gewähltem Schwierigkeitsgrad wird ein unterschiedlicher Track abgespielt, und auch das Hintergrundbild passt sich entsprechend an. Das Menü ist mit Buttons gestaltet, die ihre Farbe beim Darüberfahren ändern das gibt dem Spiel einen kleinen modernen Touch, auch wenn es insgesamt schlicht gehalten ist.

4.3 Was funktioniert, was nicht (noch)?

Das Spiel funktioniert grundsätzlich gut: Das Menü lässt sich bedienen, das Spiel startet zuverlässig, Punkte werden gezählt, und das Highscore System funktioniert ebenfalls. Die Blöcke lassen sich wie erwartet nach links, rechts und unten bewegen sowie rotieren. Auch das Nachladen eines neuen Blocks nach dem Platzieren klappt

wie geplant.
Was noch nicht ganz zuverlässig funktioniert, ist die Rotation in engen Situationen etwa wenn ein Block direkt neben einem anderen oder an der Wand rotiert werden soll. Dann kann es passieren, dass das Spiel abstürzt oder der Block in eine ungültige Position gerät. Die Methode `rotate()` in Kombination mit `block_inside()` und `block_fits()` scheint hier noch nicht alle Randfälle sauber abzudecken.
Darüber hinaus gibt es kleinere Punkte, die noch verbessert werden könnten: Die Musiklautstärke lässt sich aktuell nicht anpassen, und auch das UI (z.B. Schriftart, Buttonplatzierung oder Farben) könnte optisch noch runder gemacht werden. Der Hardmode unterscheidet sich aktuell nur durch die Geschwindigkeit und das Hintergrundbild – hier wäre es spannend, noch zusätzliche Herausforderungen einzubauen, z.B. Block Spawns an ungewöhnlicheren Positionen.

5 Zusammenfassung und Ausblick

5.1 Was haben wir erreicht?

Ein Tetris Game was sich Spielen lässt, manchmal noch kleine Bugs hat, aber an sich nicht groß das Spiel beeinflusst. Das Spiel hat sein eigenen Look und hebt sich bisschen hervor als die anderen Tetris Games die ich bisher gespielt habe. Ich finde das Spiel hat den Zweck von einem kleinen Lächeln für die Spieler*innen die es zum ersten mal starten gut erfüllt!

5.2 Was würden wir anders machen?

Den ersten beiden Sprints habe ich leider nicht wirklich ernst genommen und daraufhin kaum etwas gemacht, dass dies ein großer Fehler war und meine Planung am ende sehr schlecht war und sehr wenig funktoniert wie ich es erwartet habe wurde mir erst am späteren zeitpunkt meines Projektes klar.

5.3 Ideen für Erweiterungen

Für mein Tetris spiel habe ich noch so paar Ideen, die das Spiel optisch und vom Spielgefühl von anderen Tetris Spielen abheben würde. Z.b mit der Katzen Thematik weiter gehen, und die Blöcke in verschiedenen Katzen formen erscheinen lassen. Die Sounds für die Rotation vielleicht mit einem "miau"hinterlgegen. So wie für andere Sounds, die vielleicht mehr Katzen thematisch gestalten. Außerderm würde ich gern bei den verschiedenen Schwierigkeiten nicht einfach die Geschwindigkeit der runterfallenden Blöcke ändern, sondern z.B auch die Position, jedes mal anders von wo Sie runter fallen, oder dass Sie manchmal einen doppelten "tick"fürs down moven bekommen. Außerdem soll das Spiel noch konstant sein.

A Anhang A: Lessons Learned from the Lecture

A.1 Hilfreiche Konzepte aus der Vorlesung

Einige Konzepte aus der Vorlesung zu objektorientierter Programmierung (`OOP`) haben mir besonders beim Aufbau meines Projekts geholfen. Ich habe das Vererbungskonzept genutzt, um die unterschiedlichen Blockformen in Tetris effizient umzusetzen. Alle Blöcke (darunter IBlock, JBlock, OBlock etc.) leiten sich von der gemeinsamen Basisklasse Block ab. Dadurch war es möglich, allgemeine Methoden wie `move()`, `rotate()` oder `draw()` nur einmal zu definieren und sie wiederzuverwenden – analog zum Beispiel mit der Superklasse Boat und den Subklassen `MotorBoat` und `SailBoat` aus der Vorlesung. Auch das Prinzip „Do one thing, do it well“ erwies sich als hilfreich, etwa bei der Trennung der Klassen `Grid, Game, Button und Music`. Jede Klasse erfüllt eine eindeutig definierte Aufgabe, was auch mit dem „Single Responsibility Principle“ aus der Vorlesung übereinstimmt.
Aus dem Testing Teil habe ich vor allem mitgenommen, wie wichtig es ist, beim Programmieren darauf zu achten, dass Code testbar bleibt. Auch wenn ich keine echten Unit Tests geschrieben habe, habe ich Funktionen wie `block_fits()` oder `block_inside()` so gestaltet, dass sie unabhängig arbeiten und leicht überprüfbar sind. Außerdem habe ich beim Debugging darauf geachtet, Randfälle (wie Kollisionen am Rand des Spielfelds oder bei der Rotation) gezielt zu testen inspiriert von „Boundary Value Analysis“ aus der Vorlesung.
Rückblickend hat mir das Wissen aus der Vorlesung geholfen, mein Projekt strukturierter und nachvollziehbarer umzusetzen, auch wenn ich manches erst im Nachhinein richtig verstanden habe.

B Anhang B: Sprint-Tagebuch

B.1 Sprint 1 (08.05.–22.05.)

Den ersten Sprint habe ich leider ziemlich schleifen lassen. Ich hatte zwar eine grobe Idee im Kopf, aber wirklich gemacht habe ich in dieser Phase noch nichts. Rückblickend hätte ich da direkt mehr investieren sollen.

B.2 Sprint 2 (22.05.–05.06.)

Den ersten Sprint habe ich leider ziemlich schleifen lassen. Ich hatte zwar eine grobe Idee im Kopf, aber wirklich gemacht habe ich in dieser Phase noch nichts. Rückblickend hätte ich da direkt mehr investieren sollen.

B.3 Sprint 3 (05.06.–19.06.)

In diesem Sprint ging es ans Eingemachte: Ich habe den Hauptteil des Codes geschrieben – das Grid, die Blöcke, das Bewegen, Punkte zählen, Kollisionen erkennen usw. Das Spiel hat in dieser Phase angefangen, wirklich Form anzunehmen.

B.4 Sprint 4 (26.06.–10.07.)

Hier habe ich das Ganze dann aufpoliert: Ich hab ein Menü eingebaut, Musik hinzugefügt, verschiedene Schwierigkeitsstufen umgesetzt und auch am UI gearbeitet. Außerdem hab ich Fehler ausgebessert und alles ein bisschen runder gemacht.

C Quellen und verwendete Materialien

- PyGame-Dokumentation: `https://www.pygame.org/docs/`
- Inspiration durch mehrere YouTube Tutorials:
  `https://www.youtube.com/watch?v=zfvxp7PgQ6c`
  `https://www.youtube.com/watch?v=GMBqjxcKogA`
  `https://www.youtube.com/watch?v=nF_crEtmpBo`
  `https://www.youtube.com/watch?v=2iyx8_elcYg`
- Vorlesungsfolien „Object Orientation“, „Testing“ und „Use Cases“
- Stack Overflow für einzelne Problemlösungen z.B zur Rotation oder Musikverwaltung