Asteroids

Elias Wahl[1]

[1] *Friedrich-Schiller-Universität Jena, 07743 Jena, Germany*

Als Abschlussprojekt interpretiere ich das bekannte Spiel „Asteroids“in meiner eigenen Version. Mit dem vom Spieler kontrollierten Raumschiff müssen möglichst viele Asteroide zerstört werden, ohne das es selbst an einem zerschellt. Zu dem bekannten Spielprinzip wurde zusätzlich ein Punkte- und Highscore-Feature eingeführt, sowie aufsammelbare Perks, die dem Spieler neue mächtige Werkzeuge in die Hand geben die Asteroidem zu zerstören.

I. HOW TO PLAY

Startet man das Spiel wird man direkt in die Action geschmissen, es geht sofort los. Gesteuert wird Ihr blaues, pfeilförmiges Raumschiff mit den Pfeiltasten. Aber Vorsicht! Gerade am Anfang bietet es sich an, sich nur langsam und gefühlvoll zu bewegen. Denn so ein Raumschiff ist gar nicht so einfach zu steuern. Am besten tippen Sie nur kurz, um Steuerimpulse zu geben. Huch, was sehen sie da? Ein Asteroid kommt direkt auf Sie zu und Sie können nicht mehr ausweichen?! Zum Glück haben Sie auch eine Kanone an Bord, die Sie mit Space betätigen können. Einfach zielen, Feuer... BAM - geht doch! Der Asteroid hat sich in Luft aufgelöst, als Sie ihn mit Ihrer Bordkanone getroffen haben. Aber Sie müssen aufpassen: Bei großen Asteroiden kann es sein, dass Sie nach dem Abschuss in mehrere kleinere Fragemente zersplittern. Das kann ganz schön nervig werden.

Schauen Sie jetzt mal nach unten rechts. Das ist die Score-Anzeige, für jeden abgeschossenen Asteroiden gibt es einen Punkt. Bei Ihnen steht da also eine Eins für Ihren ersten Asteroiden, den Sie soeben abgeschossen haben. Links daneben sehen Sie den Highscore, also der höchste Score, den Sie je geschafft haben. Bei Ihnen noch eine Null, weil es Ihr erstes Spiel ist. Aber nicht verzagen, man wird deutlich schneller besser als man denkt.

Vielleicht ist jetzt auch schon irgendwo in Ihrem Fenster ein Perk aufgetaucht. Das sind Kreise in verschiedenen Farben. Z.B. gibt es ein Perk in hellblauer Farbe. Das ist ein Schild, wenn Sie es einsammeln werden Sie dauerhaft einen Schutz haben. Bei der nächsten Kollision mit einem Asteroiden zerschellt dann dieser und nicht Ihr Schiff. Das Schild ist dann aber auch weg. Das Rainbow-Perk, welches in verschiedenen Farben blinkt, hat eine ähnliche Funktion. Fliegen Sie über dieses sind Sie für eine bestimmte Zeit (solange Sie bunt blinken) unzerstörbar und können das Asteroidenfeld etwas ausdünnen indem Sie rambomäßig die Asteroiden mit Ihrem Schiff weg rammen, um sie zu zerstören. Wenn Ihnen Ihre Bordkanone nicht genügend Wumms hat, sollten Sie nach dem Laser-Perk Ausschau halten. Der rote Kreis ersetzt Ihren normalen Schuss für eine bestimmte Zeit mit einem mächtigen Laser, der alles in der Schussbahn sofort pulverisiert. Sie haben sehr viel Glück, wenn Sie das weiße Nuke-Perk berühren können. Denn dann wird sofort eine atomare Explosion ausgelöst, die alle Asteroiden zerstört. Sie bekommen sogar kurzzeitig ein Schild, sodass Sie sich wegen der Explosion keine Sorgen um Ihr Schiff machen müssen.

So haben Sie jetzt also gerade alle Asteroiden geklärt, aber jetzt tauchen direkt wieder neue auf? Das liegt an den angepassten Spawnraten. Wir wollen nicht, dass Ihnen langweilig wird. Daher spawnen, wenn es gerade nur wenige Asteroiden gibt sehr wahrscheinlich schnell mehr. Dafür ist es aber auch so, dass kaum neue Asteroiden dazu kommen, wenn schon sehr viele da sind. Davon abgesehen wird die Spawnrate von Asteroiden auch immer höher mit der Zeit in der Sie schon im Spiel sind, um den Schwierigkeitsgrad zu erhöhen.

Klingt gut? Dann ganz viel Spaß im All und bei vielen Highscoren!

II. ÜBERSICHT ÜBER DIE MODULE

- Die Hauptklassen sind entweder direkt Teil des Spiels oder sind für den Spielfluss und die Spiellogik verantwortlich.
  - main.py: Dieses Modul bildet das Herzstück des Spiels. Mit der Klasse Game initialisiert es das Spielfenster, bestimmt die Spawnraten von Asteroiden & Perks und steuert den Spielablauf. Die Spielschleife befindet sich hier und sorgt für die kontinuierliche Aktualisierung und Darstellung des Spiels. Das Modul beendet das Spiel ebenfalls und speichert den End-Score im den Fall, dass es ein neuer Highscore ist.
  - entities.py: Dieses Modul kümmert sich mit der Klasse Entities um alle Entitäten im Spiel, die in Klassen organisiert sind. Dazu gehören das Raumschiff, Asteroiden, Projektile und Explosionen. Das Modul verwaltet die Erstellung, Positionierung und Bewegung dieser Entitäten. Die Kollisionsabfrage und -behandlung spielt eine wichtige Rolle, um Interaktionen zwischen den Entitäten zu erkennen und die Spiellogik korrekt umzusetzen. Punktevergabe findet hier ebenfalls statt.

  - asteroids.py: Die Generierung und Steuerung von Asteroiden unterschiedlicher Größe und Form gehört zum Aufgabenbereich dieses Moduls. Das Splittern von großen Asteroiden in mehrere kleine Fragmente sorgt für zusätzliche Herausforderung und Dynamik im Spiel.
  - spacecraft.py: Dieses Modul steuert mit der Klasse Spacecraft das Raumschiff, welches vom Spieler durch die Pfeiltasten gelenkt wird. Die Steuerung beinhaltet Beschleunigung, Rotation und das Abfeuern von Geschossen. Durch das Aufsammeln von Perks kann das Schiff Spezialfähigkeiten wie ein Schutzschild, einen Laser oder temporäre Unverwundbarkeit erreichen, um dem Spieler zusätzliche taktische Möglichkeiten zu bieten.
  - projectiles.py: Zwei verschiedene Arten von Projektilen mit unterschiedlichen Eigenschaften und Klassen werden in diesem Modul in der Oberklasse Projectiles erwaltet. Die Standardwaffe - einfache Schüsse mit begrenzter Lebensdauer der Klasse SimpleShot, können durch einsammeln des Laser-Perks temporär zu einem instantanen Laser upgegraded werden.
  - perkz.py: Perks (Power-Ups) tauchen zufällig im Spielfeld auf und können vom Spieler eingesammelt werden. Verschiedene Arten von Perks, wie ein Schutzschild, Laser oder eine Nuke, bieten dem Spieler temporäre Vorteile und taktische Möglichkeiten. Das Modul verwaltet das Spawnen und die Anzeige der Perks.
  - explosions.py: Dieses Modul ist für die spektakulären Explosionen im Spiel verantwortlich. Es erstellt und animiert Explosionen mit der Klasse Explosions in verschiedenen Größen je nach zerstörtem Asteroiden. Auch das mächtige Perk "nuke" wird mit diesem Modul druch die Klasse Nuke dargestellt.

- Die Hilfsklassen sind nicht direkt Teil des Spiels, aber die Hauptklassen fallen für ihre Konstruktion und ihre Methoden auf sie zurück.

  - moving_objects.py: Die Basisklasse MovingObj in diesem Modul definiert grundlegende Eigenschaften für alle beweglichen Objekte im Spiel, i.e. translatorische Geschindigkeit, Rotationgeschwindigkeit und Farbe. Diese Eigenschaften werden von den Klassen für das Raumschiff, Asteroiden und Projektile geerbt und ermöglichen eine weitgehend einheitliche Aktualisierung und Bewegung dieser Objekte.
  - polygon.py: Die uns gestellte Klasse Polygon vereinfacht das Malen verschobener und verdrehter Polygone deutlich und antwortet auf die Frage, ob ein bestimmter Punkt innerhalb eines Polygons liegt.
  - point.py: Die Klasse Point besteht aus einem Paar floats und Rechenoperationen. Sie wird vor allem als Koordinate im Ortsraum verwendet, aber auch für die translatorische Geschwindigkeit von Objekten. Die Methoden der Klasse Polar vereinfachen das Rechnen mit Polarkoordinaten in der Klasse Point.

III. GROBER ABLAUF

Wenn ein neues Spiel gestartet wird, wird in main.py die Variable game der Klasse Game initialisiert. Bei der Initialisierung wird auch Entities() eingerichetet. Wiederum dadurch wird auch das erste mal Score, Spacecraft & Asteroids aufgerufen. Das heißt bei Score, dass der Highscore aus der highscore.txt geladen wird. Dem Spacecraft werden seine Startwerte ohne Geschwindigkeit und ohne Perks in der Mitte des Canvas zugewiesen. Über Asteroids werden direkt die ersten Asteroid gespawned, die schon im ersten Frame existieren.

Die Leinwand wird dann über game.new_canvas eingerichtet und das Spiel erreicht den Spiele-Loop in game.run(). Als erstes wird hier in random_spawn() die aktuelle Spawn-Rate für Asteroiden berechnet und mit dieser Wahrscheinlichkeit ein neuer Asteroid erstellt. Auch Perks haben hier eine Chance zu spawnen. Wenn sie das tun, dann mit zufälliger Position und zufälligem Perk-Typ.

Als nächstes werden über self._entities.update() alle entities geupdated. Das heißt zuerst, dass gecheckt wird, ob der Spieler gerade, die Spacetaste drückt, um einen Schuss abzugeben und ob der Cooldown seit dem letzten Schuss schon wieder auf Null ist. Wenn das Beides der Fall ist wird Entities das aktive Projectile angehängt. Danach geht das Programm durch jedes entity von Entities und stellt zuerst sicher, dass alle entities mit status gleich -1 entfernt werden. Danach werden bei jedem enitiy Position und Rotation den entsprechenden Geschwindigkeiten entsprechend geändert werden und die Stati aktualisiert.

Schlußendlich werden alle Asteroiden und Perks auf Kollisionen getestet. In der Funktion entities.is_interacting wird performance-effektiv geprüft, ob es irgendwelche Berührungen zwischen den Hitboxen interaktions-relevanter Klassen gibt und im Fall die entsprechenden Aktionen durchgeführt. Trifft ein Projectile oder ein Spacecraft mit Shield auf einen Asteroiden, so wird der zerstört und löst eine Explosion aus, die aber nicht weiter interagiert. Sammelt das Spacecraft ein Perk ein, verschwindet das Perk und der Effekt wird entweder auf das Spacecraft übertragen oder direkt ausgeführt.

Am Ende eines Frame-Cycles werden alle entites mit self._entities.draw() gemalt und über std.show(1000/p.FPS) angezeigt. Sollte das Spacecraft durch die Kollision mit einem Asteroiden zerstört werden, endet der Game-Loop und über game.end() wird der Game-Over-Screen angezeigt mit der erreichten Score-Zahl, die hervorgehoben wird, wenn sie ein neuer Highscore ist. Dem Spieler wird die Möglichkeit gegeben mit Enter weiterzuspielen oder mit ESC das Spiel zu verlassen.

IV. HIGHLIGHTS

Wir schauen uns nun noch zwei interessante Codestellen etwas genauer an und erläutern die Idee dahinter.

A. Laser Trefferberechnung

Zuerst möchte ich auf die Methode in_beam von der Klasse Laser() eingehen. Die Funktion berprüft effizient, ob ein Asteroid von einem Laser getroffen wird. Der Algorithmus nutzt Techniken, um die Rechenleistung zu optimieren und trotzdem eine sichere Antwort geben zu können.

Zuerst wird die Steigung des Lasers berechnet. Je nach Steigung wird die x- oder y-Koordinate des Asteroiden verwendet, um einen Punkt auf der Laserlinie zu berechnen. So wird sicher gegangen, dass ein Punkt auf der Laserlinie gefunden wird, der sehr nah am Punkt ist. Der Abstand zwischen diesem Punkt und der Position des Asteroiden wird mit einem Schwellenwert verglichen. Liegt der Abstand unterhalb des Schwellenwerts, wird eine genauere Prüfung durchgeführt. Die Funktion right_direction wird zudem verwendet, um sicherzustellen, dass der Laser auf den Asteroiden zeigt und nicht nicht in die entgegengestzte Richtung.

In einem Bereich um den berechneten Punkt werden 10 Punkte auf der Laserlinie generiert. Für jeden Punkt wird geprüft, ob er innerhalb des Asteroiden liegt. Trifft dies zu, wird die Funktion True zurückgeben, was bedeutet, dass der Asteroid vom Laser getroffen wurde.

Die Verwendung der Steigung ermöglicht es, die relevanten Koordinaten des Asteroiden effizient zu berechnen. Der Schwellenwert filtert schnell Punkte heraus, die weit vom Laser entfernt sind. Die genaue Prüfung beschränkt sich auf einen kleinen Bereich um den berechneten Punkt. Die Verwendung von 10 Punkten statt einer größeren Anzahl reduziert den Rechenaufwand, während die Genauigkeit erhalten bleibt.

Diese Methode ist deutlich effizienter als die direkte Berechnung der Schnittpunkte zwischen Laser und Asteroid. Die Laufzeit ist nahezu konstant und unabhängig von der Größe des Asteroiden.

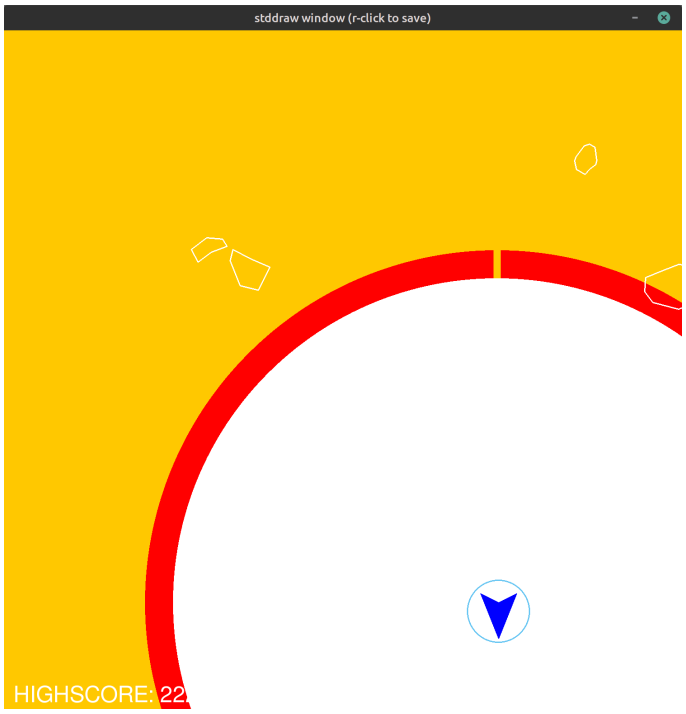
B. Asteroiden Initialisierung

Die Generierung der Eckpunkte des Asteroiden erfolgt leichter in einem Polarkoordinatensystem. Jeder Eckpunkt des Asteroiden wird durch seinen Abstand (length) vom vorherigen Punkt und den Winkel (alpha) im Verhältnis zum vorherigen Punkt definiert. Die Methode random_asteroid_shape von der Klasse Asteroid legt die Anzahl der Ecken für den Asteroid zufällig zwischen 5 und 13 fest.

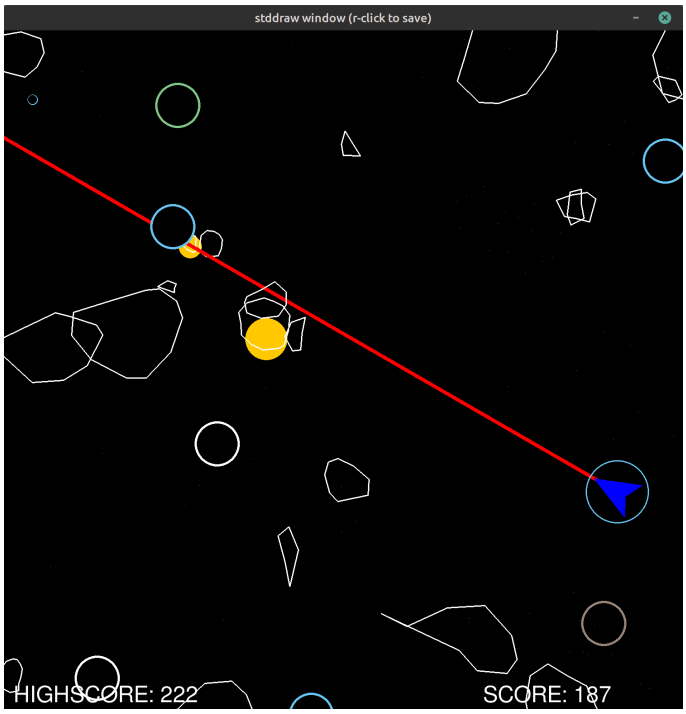
Der erste Eckpunkt des Asteroiden wird auf den Ursprung (0, 0) gesetzt. Der nächste Punkt wird wie folgt erzeugt: Ein zufälliger Abstand zwischen 40% und 140% der Kantenlänge des Asteroiden wird ausgewählt. Ein Winkel relativ zum vorherigen Punkt wird berechnet. Zunächst gibt es eine gleichverteilte Basisrichtung im Kreis, zu der je nach Eckpunkt ein gewisser Bereich an Zufälligkeit hinzugefügt wird.

Der erzeugte Punkt in Polarkoordinaten wird in kartesische Koordinaten umgewandelt. Durch relative Angabe des Punktes (Abstand und Winkel zum vorherigen Punkt) wird der nächste Eckpunkt dem Polygon hinzugefügt.

Beim letzten Punkt wird der Abstand so gewählt, dass dieser etwa auf halber Distanz zum Ursprung liegt. Dies soll verhindern, dass der letzte Punkt sehr weit herausragt und unnatürlich lange Spitzen bildet. Der Winkel wird so gewählt, dass er in etwa in Richtung des Ursprungs zeigt, um das Schließen der Asteroidenform zu ermöglichen.



(a) Das Perk Nuke in Action.



(b) Das Perk Laser in Action.