FACH
HOCHSCHULE
LÜBECK
University of Applied Sciences

Webtechnik Projekt

Raumffisch

Inhaltsangabe:

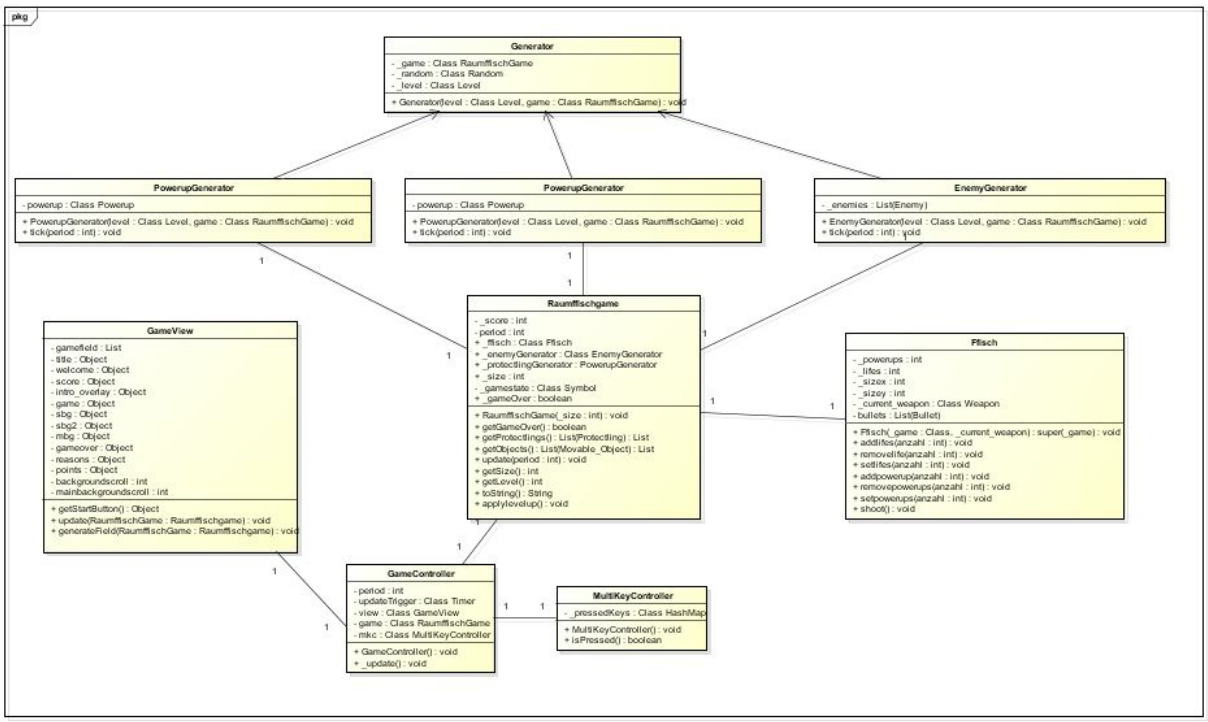
1. Einleitung

Eine Dokumentation über die Entwicklung des Webtechnik-Projekts "Raumffisch", begleitet von Herrn Prof. Dr. N. Kratzke. Zur Durchführung wurde das "Snake Dart" Spiel von Herrn Prof. Dr. N. Kratzke als Grundlage herangezogen(bei Github geforked) und von Johannes C. Gosch, Kim O. Schweikert und Martin Piontek in soweit modifiziert, dass sich ein Spiel mit eigenem und neuem Spielkonzept entwickelt hat. Ein so genanntes “Side-Scrolling-Space-Shooter-Spiel”.

2. Spielkonzept

Das Spiel "Raumffisch" ist ein "Side-Scrolling-Space-Shooter-Spiel". Der Spieler beginnt sein Abenteuer am linken Spielfeldrand mit 3 Leben und bahnt sich seinen Weg durchs All. Er kann dabei aufwärts, abwärts, nach rechts oder nach links fliegen. Da die Hauptfigur ein "Ffisch" ist wird man stets von einem Stichling(Protectling) begleitet, dieser bewegt sich abwechselnd auf und ab und langsam in Richtung des linken Spielfeldrandes. Schafft er es, diesen zu passieren, erhält man ein Leben und 10 Scorepunkte; man kann jedoch nur maximal 5 Leben haben. Die Nebenaufgabe ist es also, den Protectling zu beschützen und zu verteidigen. Gegner tauchen immer wieder am rechten Spielfeldrand auf. Diese sollte man mit Hilfe eines Schusses zerstören. Jeder Abschuss bringt Punkte und treibt den Score nach oben. Wird man von einem Gegner getroffen verliert man ein Leben, so lange bis man keines mehr besitzt. Dann endet die Reise. Stirbt ein Stichling(Protectling), sei es durch Eigenbeschuss oder Kollision eines Feindes, verliert man Scorepunkte. Zur Unterstützung gibt es für den "Ffisch" einige PowerUps, die es ihm ermöglichen, verschiedene Waffentypen zu erhalten. Ein Level ist dann überwunden, wenn man eine bestimmte Strecke zurückgelegt hat. Auf dem höchsten Level endet das Spiel erst dann, wenn man selbst stirbt. Das Ziel des Spiels ist es, den höchsten Score zu erreichen.

3. Architektur



4. Model

Im Model werden alle Spielinhalte und Zustände gehalten und überwacht. Es gibt bewegliche Objekte, welche in einer Liste _objects gespeichert werden. Diese besteht aus den Instanzen der Klassen Protectling, Ffisch, Enemy, Powerups und Bullet.

Der Spielstatus wird per boolean festgehalten. Dazu wird _gameOver beim Start false gesetzt. Verfügt die Spielfigur über keine weiteren Leben wird der Wert true gesetzt und das Spiel beendet.

Ein Konstruktor erstellt ein neues Spiel mit feststehender Spielfeldgröße von 80x80, einer steuerbaren Spielfigur am linken Spielfeldrand, einem Protectling der sich vom unterem Spielfeldrand nach oben bewegt und den Generatoren zur Erstellung der Feinde und der PowerUps, welche sich vom rechten Spielfeldrand nach links durch das Spielfeld bewegen. Die Häufigkeit dieser Objekte ist an das Levelkonzept gebunden. Je höher das Level, desto mehr Gegner aber weniger PowerUps. Die Protectlings, welche kontinuierlich nachgeneriert werden, da sie vom Gegner oder den eigenen Geschossenen zerstört werden können, werden ebenfalls von einem Generator erstellt.

Die Geneartoren überwachen den Lebenszyklus und stoßen das Verhalten ihrer zu verwalteten Objekte vie tick()an.

Die Methode void applylevelup() informiert den Spieler darüber ein Level absolviert zu haben und sorgt dafür, dass das nächste Level erstellt wird. (Johannes fragen)

Die Zustandsverändernde Funktion unseres Models ist die void update(int period) Methode, sie übermittelt die Zustände in einem bestimmten Takt an die View, welcher im int period festgehalten wird. An dieser Stelle wird auch die void applylevelup() Methode aufgerufen.

Der Ffisch hält eine Instanz einer das Interface Weapon implementierenden Klasse. Sie legt die Waffenart fest. Die Waffenart kann durch das Einsammeln von Powerups verändert werden. Sämtliche Waffen sind in der Datei weapons.dart implementiert. Sie erstellen eine Liste von Instanzen der das Bullet-Interface, bestimmt für die Geschosse und der Verhalten, implementierenden Klassen, wenn ihre Methode shoot() aufgerufen wird und geben diese Zurück, um sie der Objektliste hinzufügen zu können. Die Bullets sind in bullet.dart implementiert.

Die sich über das Spielfeld bewegenden Gegner sind in enemy.dart enthalten und erweitern, wie jedes bewegliche Objekt die Klasse Movable_Object.

5. View

Die View visualisiert alle Inhalte des Models im Browser. Aus Performancegründen erhält sie ein zweidimensionales Array mit allen Referenzen der Tiles des optischen Spielfeldes(td-Elemente innerhalb einer Tabelle).

Aufgrund der Aufwendigkeit zur Erstellung aller CSS-Klassen der anzuzeigenden Objekte entwickelten wir das Toolkit “Odysseeanemone”. Dies erstellt die CSS-Klassen zu den Tiles jeder Objektgrafik.

Um den Eindruck einer Fortbewegung unserer Figuren durchs All zu simulieren nutzten wir die Technik des Parallax-Scrolling. Dies verleiht dem Spiel zudem eine “räumliche” Darstelllung.

Über die Anweisung keinen Ton oder Sound zu implementieren haben wir uns hinweggesetzt, da wir der Meinung sind, dass ein Shooter ohne Sound kein richtiges Spielgefühl übermittelt.

6. Controller

Der Controller überwacht die Tastatureingabe des Spielers mittels einer Instanz der Klasse MultiKeyController. Dieser hält eine Liste aller gedrückten Tasten, die an das Model übergeben werden.

Der Controller gibt den Takt des Spiels vor. Dadurch wird gewährleistet, dass sich die View bis zum Spielende regelmäßig updated um die Objekte des Models zu visualisieren.
Im Model werden die Generatoren aufgerufen um die jeweiligen Objekte zuerstellen und deren Verhalten anzustoßen (Tick Methode).

7. Zusammenfassung und Ausblick

Die Entwicklung des Spiels verlief über 5 Wochen, von der 24.-28. Kalenderwoche des Jahres 2015. In dieser Zeit setzten wir unsere Ideen und die Vorgaben von Prof. Dr. N Kratzke in das Spiel um. Jedes Level wird von Links begonnen. Die Gegner tauchen Zufällig am rechten Spielfeldrand auf, deren Häufigkeit und der der PowerUps, sind abhängig vom Level. Zurzeit verfügt das Spiel über Zwei Level. Das Erste ist so eingestellt wie es später auch sein soll, das Zweite hat eine erhöhte Vorkommensrate der PowerUps, damit alle einmal eingesammelt werden können.

Das Balancing der jeweiligen Level müssten noch angepasst werden. Die meisten implementierten