**Protokoll**

**Space Invaders Programm mit Python**

**Europaschule**

**Schulzentrum SII Utbremen**

**Fach/LF:** LF07

**Klae/Kurs:** ITA20

**Name/Gruppe:** Omar/Michael

**Datum:**

06.10.2021

**Inhaltsverzeichnis**

Seite

1. Beschreibung der Aufgabenstellung 3

2. Benutzeranleitung 3

3. Programmbeschreibung 3

4. UML Diagramm 5

5. Erkenntnisgewinn DAKI 5

5.1 Drop 5

5.2 Add 5

5.3 Keep 6

5.4 Improve 6

6. Abbildungsverzeichnis 6

7. Quellenverzeichnis 6

7.1 Links 6

# Beschreibung der Aufgabenstellung

Es soll das Spiel „Space Invaders“ mit Hilfe von Pygame programmiert werden. Im Spiel steuert der Benutzer einen Panzer und muss auf die herunterkommenden Aliens schießen. Dabei kann der Benutzer sich hinter Barrieren verstecken, die nach und nach von den Aliens zerstört werden. Dies soll mit dem Verfahren der objektorientierten Programmierung geschehen. Zudem werden verschiedene Klassen vorgegeben die implementiert werden sollen.

# Benutzeranleitung

Beim starten des Spiels wird Ihnen das Anfangs-Menü gezeigt. Dort können Sie das Menü mit der Maus, den Pfeiltasten und der Enter-Taste steuern. Im Menü können sie die Anzahl der Gegner einstellen und das Spiel starten oder beenden. Wenn Sie das Spiel gestartet haben können sie mit Pfeil-Links und Pfeil-Rechts den Panzer bewegen und mit der Leertaste Schießen. Sie kehren automatisch ins Menü zurück, wenn die Gegner es geschafft haben unten anzukommen oder Sie alle Gegner eliminiert haben. Falls Sie das Spiel vorzeitig beenden möchten, steht Ihnen dafür die Escape-Taste zur Verfügung.

# Programmbeschreibung

Zunächst einmal werden alle erforderlichen Module ganz am Anfang importiert. Die Klasse *Settings* dient zum Konfigurieren von einigen Optionen im Spiel. Hier wird auch das Menü des Spiels erstellt anhand des Moduls *Pygame-Menu* erstellt. Die Klasse *Ship* erbt von *pygame.sprite.Sprite* und es erstellt ein steuerbares Schiff im unteren Bereich des Spielfeldes, welches mit den Pfeiltasten gesteuert werden kann. Der Score und Highscore werden mit den Methoden *collision\_detection* und *set\_high\_score* aktuell gehalten. In *Shield* sind die Eigenschaften der Barrieren festgelegt.

Die Methode *spawn\_shield\_group* erstellt eine Gruppe von vier Hindernissen, welche vor dem Schiff platziert sind. Die Sprites von diesen werden nach jedem Treffer von einem Laser aktualisiert, sei es der von dem Schiff oder von den Gegnern. Die Hindernisse werden vom Spielfeld gelöscht, wenn sie 3-mal von einem Laser getroffen wurden.

In der Klasse *Aliens* werden je nach Benutzereingabe 10, 20, 30 oder 50 Aliens auf dem Spielfeld gespawnt. Diese werden mit verschiedenen Methoden bewegt und animiert.

Der Score wird mit der Klasse *Score* aktualisiert und gespeichert. Bei jeden eliminierten Gegner erhält man 10 Punkte. Mit einer weiteren Klasse wird der Highscore festgehalten. Sie wird mit den Methoden *define\_highscore* und *save\_highscore\_in\_external\_file* in eine externe Datei geladen und aufgerufen.

Die Klasse *Laser* umfasst der Laser von den Gegnern und dem Schiff. Für Schiff ist der Laser ein Blitz und für die Gegner ist es eine schwarze Kugel.

In *Game* wird die Interaktion zwischen den Klassen, wie z. B. die Bewegung von den Lasern und die Kollisionserkennung, definiert. Der Laser vom Schiff wird nach einer Kollision nicht gelöscht, sondern wieder zu seiner Startposition zurückgesetzt und nach einem Drück auf die Leertaste wieder abgefeuert. Bei den Gegnern ist das ähnlich, es schießt bloß jedes Mal ein zufällig ausgewählter Gegner eine Kugel.

In *main()* wird zunächst einmal pygame initialisiert und alle pygame erforderliche Syntax angegeben. Die Objekte für Schiff, Aliens usw. werden hier definiert.

In der Gameloop werden zuerst die Tastendrücke verarbeitet. Die Kollisionserkennung und die Bewegung von den Lasern erfolgt gleichzeitig mit den Methoden *detect\_laser/bullet\_collision*. Die Aliens werden animiert und der Score, Highscore und Leben werden Variablen zugewiesen. Anschließend wird alles in der richtigen Reihenfolge auf dem Bildschirm gemalt und das Spielfeld wird aktualisiert. Schließlich werden die Abläufe pro Sekunde mit der Funktion *clock.tick()* ausgeführt.

# UML Diagramm

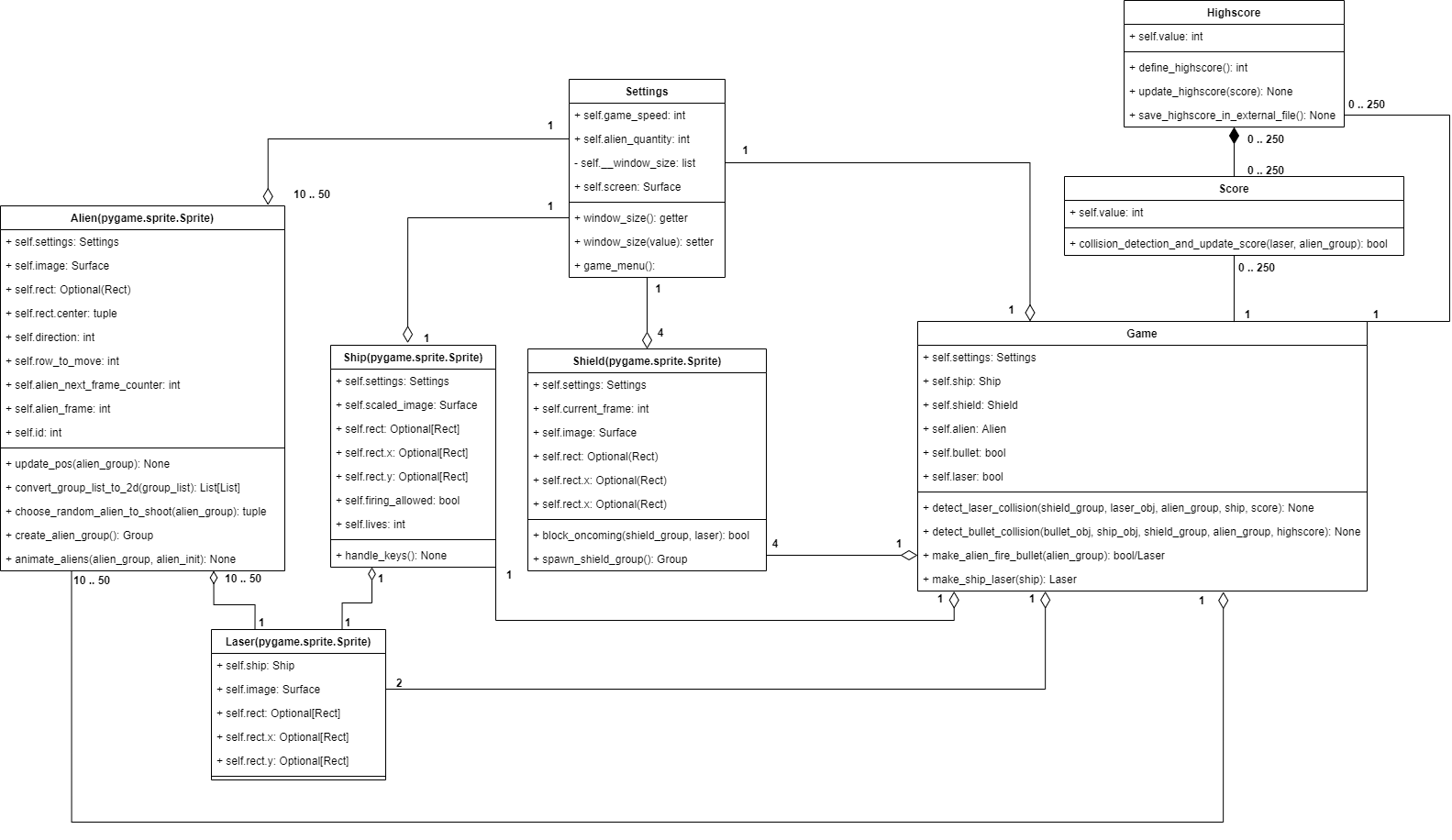


Abbildung : UML-Diagramm in diagrams.net

# Erkenntnisgewinn DAKI

## Drop

Wir werden keine öffentlichen Variablen bei der Gruppenprogrammierung mehr verwenden, da die Gefahr besteht, dass einer die Variablen verändert und das Programm für mehrere Programmierer bis auf weiteres unbrauchbar macht.

## Add

Objektorientiertes Programmieren wollen wir weiterhin anwenden. Wir sehen viele Vorteile und eine Struktur, die die semantische Lücke zwischen uns Menschen und das Programmieren verkleinert. Es gibt uns eine Struktur, mit dem man schneller und besser arbeiten kann, als wenn man es nicht anwendet.

## Keep

Ein UML-Diagramm möchten wir definitiv weiterhin verwenden. Wir finden, es ist sehr sinnvoll, dass man vor dem Programmieren seine Programmstruktur mit einem UML-Diagramm dokumentiert und schließlich mit dem Endresultat vergleicht. Somit kann man in der Zukunft immer effizienter arbeiten, wenn man das Diagramm immer präziser gestaltet.

## Improve

Die Implementierung von Datenkapselung ist uns wegen Unerfahrenheit nicht gelungen, wie wir es gehofft hatten. Wir werden künftig Attribute oder Methoden, worauf keiner Zugriff haben sollte, privat machen und mit „getter“ und „setter“ zugänglich machen.

# Abbildungsverzeichnis

[Abbildung 1: UML-Diagramm in diagrams.net 5](#_Toc85303052)

# Quellenverzeichnis

## Links

Name: “Space Invaders”

Link: https://villains.fandom.com/wiki/Space\_Invaders

Anmerkung: Für die Sprites von den Gegnern verwendet. Der Hintergrund der Gegner wurde mit Paint3D entfernt.

Name: „4K Star Wallpapers“

Link: <https://wallpaperaccess.com/4k-star>

Anmerkung: Der Hintergrund des Spiels.

Name: „Space Invaders“

Link: https://www.spieleklassiker.com/spiel/Space+Invaders

Anmerkung: Für die Barriere verwendet.

Name: „Laser Cannon“

Link: https://spaceinvaders.fandom.com/wiki/Laser\_Cannon

Anmerkung: Der Sprite vom Schiff