Dokumentation Projekt Asteroids

Ersteller:	Birger Möllering
Ausbildungsberuf:	Fachinformatiker Anwendungsentwicklung
Ausbildungsbetrieb:	Third Element Aviation GmbH
Ausbilder:	Stephan Berkowitz sb@3rd-element.com
Erstellt am:	1.5.2020

Inhaltsverzeichnis

1. Darstellung der Projektphasen

Im ersten Teil der Dokumentation beschreibe ich die Ausgangslage, die Anforderungen an das Projekt gefolgt von der Planung und der tatsächlichen Durchführung. Abschließend beschreibe ich die technischen Zusammenhänge sowie, wie die Qualitätssicherung realisiert ist.

1.1 Ausgangssituation

Aufgrund meines Quereinstiegs in die Projektarbeit, bin ich alleine für alle Bereiche der Projektentwicklung zuständig. Das Spiel Asteroids wähle ich, da es mir in der zur Verfügung stehenden Zeit, machbar erscheint.

1.2 Anforderungsdefinition

Ziel des Projekts ist die Erstellung eines Asteroid-Klons mit begrenztem Umfang.
Der hauptsächliche Nutzen dieses Projekts ist es, mehr über die Planung und die Organisation von Softwareprojekten zu lernen.

Die Projektspezifikation lautet wie folgt:
„Es gibt ein Raumschiff das mit den Pfeiltasten gesteuert werden kann. Mit den Pfeiltasten Rechts und Links kann das Raumschiff gedreht werden. Mit der Pfeiltaste Hoch kann beschleunigt werden und mit der Space Taste kann geschossen werden. Während des Spiels fliegen Asteroiden durch das Spielfeld. Wenn das Raumschiff von einem Asteroiden berührt wird, ist das Spiel zu Ende und der Spieler bekommt die Spielzeit präsentiert, die das Raumschiff überlebt hat.“

1.3 Planung

Bei der Projektplanung lege ich den Fokus auf das Ressourcen- und Zeitmanagement.
Die Projektplanung läuft in vier Schritten ab:
1. Wahl des Projekts, mit zeitlicher Machbarkeit als Hauptfaktor.
2. Auflistung der zu verrichtenden Aufgaben.
3. Einteilung in übergeordnete Planungspunkte.
4. Zuordnung von Zeitumfängen zu Aufgaben.
Der Projektplan mitsamt Gantt-Diagramm befindet sich im Anhang (siehe 3.1).

Auch der Projektplan gliedert sich in vier Programmpunkte:

Beim ersten Punkt, der Ressourcen Evaluation, geht es sowohl um die Frage welche Ressourcen benötigt werden um das Projekt erfolgreich abzuschließen als auch um deren Eignung.
Bei der Wahl von Python als Sprache zum Verwirklichen des Projekts gab es zwei ausschlaggebende Punkte. Erstens ist Zeit meine knappste Ressource und Python ist eine Sprache, in der ich schnell Ergebnisse erzielen kann und zweitens ist es für dieses Schulprojekt unnötig, den Quellcode gegenüber anderen zu verschleiern. Die Wahl der zu verwendenden Bibliotheken fiel auf pyglet und pymunk. Pyglet ist eine „pythonische“ Fenster und Multimedia Bibliothek, die ich schon lange kenne und pymunk ein wrapper für die 2D Physik-Engine chipmunk, für die ich mich entschieden habe, weil sie viele Möglichkeiten eröffnet das Spiel weiterzuentwickeln. Als Entwicklungsumgebung benutze ich aus Gewohnheit tmux und vim im Terminal und zum Erstellen des Klassendiagramms Drawio, weil wir in der Firma Drawio verwenden und ich den Umgang mit Drawio besser erlernen möchte.
Die Ressourcen Evaluation schließt mit dem „Milestone 1“ ab, welcher durch einen vollständigen Ressourcenplan erreicht ist. Der Ressourcenplan befindet sich im Anhang (siehe 3.2)

Beim zweiten Punkt, der Programmplanung, geht es um die vorab Modellierung der Software.
In diesem Fall entschied ich mich für ein sogenanntes Klassendiagramm, dessen Fertigstellung den „Milestone 2“ darstellt. Das Klassendiagramm befindet sich im Anhang (siehe 3.3).

Der dritte Punkt, die Implementation, ist eine Aufgabe, die sich schwer vorab in Unterpunkte gliedern lässt. Die Implementation der vorab geplanten Komponenten, deren Verknüpfung und das Testen der Funktion sind ein ständiger interaktiver Prozess, der sich meiner Erfahrung nach auch nur interaktiv planen lässt. Der dritte Meilenstein ist der funktionierende Asteroids-Klon mit allen unter 1.2 spezifizierten Funktionen. Der Quellcode ist online unter https://github.com/moeb/asteroids zu finden.

Beim vierten Punkt, der Präsentation, ist wie sie zu erfolgen hat unklar. Entsprechend ist sie schwierig zu planen. Die Unterpunkte spiegeln meine Unsicherheit wieder. Letztendlich kann die Präsentation erst dann richtig ausgearbeitet werden, wenn Ergebnisse aus den vorherigen Punkten vorliegen.

1.4 Durchführung

Von Anfang an nahm die Organisation und das Lernen über die Organisation sehr viel mehr Zeit in Anspruch, als die eigentliche Implementation. Sowohl herauszufinden was gefordert ist, als auch die Beschäftigung mit dem Gantt-Diagramm forderten vorab viel Zeit. Nach mehreren Versuchen mit Excel und verschiedenen Programmplanungsprogrammen, entschied ich mich schlussendlich für die für mich einfachste Alternative, einer Vorlage für LibreOffice Calc. Bei sämtlichen Gantt-Diagramm-Programmen waren Tage die kleinste Zeiteinheit, so auch bei der verwendeten Vorlage. Um dennoch einen Wasserfalleffekt zu erzielen, entschied ich mich dafür, die Aufgaben über eine ganze Woche zu strecken. Was als Dokumentation gefordert war, habe ich am Anfang übersehen. Ansonsten wäre diese ein eigener Programmpunkt geworden.

Die tatsächliche Abarbeitung der ersten drei Programmpunkte fand fast vollständig an den Tagen Montag und Dienstag statt.

Die Ressourcenevaluation nahm speziell wenig Zeit ein, weil mir viele Bibliotheken und Software-Tools bereits bekannt waren. Entsprechend brauchte ich nicht länger als eine halbe Stunde um den ersten Meilenstein zu erreichen. Ungeplant war sowohl die Beschaffung der Ressourcen, als auch die Einrichtung der Softwaretools. Der Zeitaufwand lag ebenfalls bei ca. 30 Minuten zzgl. einer weiteren halben Stunde für das Erstellen der für das Spiel notwendigen Grafiken abzüglich der Grafik für das Geschoss. Dieses war zu diesem Zeitpunkt noch nicht eingeplant und ich war mir noch nicht im Klaren darüber, wie Geschosse im Spiel funktionieren sollten.

Der nächste Schritt war die Programmplanung. Diese begann für mich nicht wie geplant damit ein Diagramm zu erstellen, sondern damit mir vorzustellen, wie das Programm funktionieren soll. Diesen Schritt kann ich nicht auf einen klaren Zeitraum begrenzen. Stattdessen begann er in dem Moment, in dem ich das Projekt ausgesucht habe. Dennoch halte ich die Erstellung eines Diagramms für äußerst hilfreich, sowohl zwecks der inneren Ordnung, als auch der Dokumentation. Davon abgesehen war der erste Versuch ein Diagramm zu erstellen nicht zu meiner Zufriedenheit. Dennoch war es als einfaches Anfangsmodell durchaus hilfreich. Das Resultat befindet sich im Anhang (siehe 3.4). Das Klassendiagramm (3.3) entstand erst nach der ersten funktionierenden Version und war nützlich dabei, einige Zusammenhänge zu vereinfachen und somit zu verbessern.

Der dritte Schritt war die Implementation. Diese war dank der Planung größtenteils unkompliziert. Am meisten Zeit floss in Aufgaben die mit der Physik-Engine pymunk zu tun hatten. So verbrachte

ich bestimmt eine Stunde damit herauszufinden, wie die Rotationsgeschwindigkeit des Raumschiffes zu setzen ist und eine weitere Stunde um die Kollisionsdetektion zu verstehen. Insgesamt schätze ich die reine Implementationszeit auf ungefähr 10 - 12 Stunden.

1.5 Darstellung fachlicher und technischer Zusammenhänge und Funktionsweisen

Beim Planen und Implementieren des Projekts orientiere ich mich am „Model-View-Controller“ Entwurfsmuster (MVC). Dabei sind die Modelle Klassen aus der Physik-Engine pymunk
und der View basiert auf Sprites, Labels und Batches von pyglet. Während des Schreibens der Dokumentation ist mir aufgefallen, wie das Programm tatsächlich zusammengesetzt sein sollte, und welche Teile des Programms welche Zuständigkeiten haben sollten. So ist es derzeit so,
dass ich eine eigene Modell-Klasse angelegt habe, von der die Modelle Spaceship, Bullet und Asteroid abgeleitet werden. Das halte ich für unnötig. Die von Pymunk zur Verfügung gestellten Objekte allein reichen bereits aus, um den Modellteil in den Controllern zu bilden. Anstatt das ein AsteroidSpammer als Controller existiert, sollte es einen AsteroidController pro Asteroid geben und das spammen (was spawning/spawnen heißen sollte) in der main loop bzw. einem geschedulten Callback geschehen. Anstatt dass der Player sowohl das Spaceship als auch die Bullets kontrolliert, sollte der Player nur BulletController spawnen. Derzeit werden bei der „World“ sowohl das Modell als auch der Controller zusammen registriert, dabei ist das unnötig, wenn der Controller direkt pymunk Objekte verwendet. Zuletzt sollte das Kollisionshandling von der World an die betroffenen Controller weitergegeben werden anstatt durch separate Callbacks behandelt zu werden.

1.6 Kontrolle / Test

Automatisiertes Testen als auch das Arbeiten mit Unit-Tests überstieg den Rahmen meiner Möglichkeiten für dieses Projekt. Ansonsten teste ich natürlich immer wieder während des Entwicklungsprozesses als auch vor der Veröffentlichung. Damit das gut funktioniert, verwende ich einen „development“ Branch und einen „master“ Branch in dem Git Repository. Dabei wird der development Branch nur dann in den master Branch gemergt, wenn er eine auf Funktionsfähigkeit getestete Version enthält. Arbeiten auf dem „master“ Branch vermeide ich wenn möglich vollständig.

2. Reflexion der Ergebnisse und Zielerreichung

Im zweiten Teil der Dokumentation geht es darum zwischen der Projektspezifikation und dem gewünschten Ergebnis zu vergleichen und zu reflektieren was gut geklappt hat und was nicht und warum.

2.1 Review des Produkts / Qualitätssicherung

Die vordefinierte Projektspezifikation wurde vollständig erfüllt. Es gab einige undefinierte Szenarien, für die ich während des Projekts vorläufige Lösungen gefunden habe. Beispielsweise dazu, was passiert, wenn das Raumschiff den Bildschirm verlässt. Alles in allem ist das Projekt ein guter Anfang, auf den weiter aufgebaut werden kann.

Es gibt viele mögliche Weiterführungen für das Projekt von denen die mir persönlich wichtigste ist, die in Punkt 1.5 beschriebenen Änderungen durchzuführen, da ein festes Fundament die Grundlage für einen hohen Turm ist. Weitere mögliche Fortführungen beinhalten Hintergrundmusik, Hintergrundbilder, Animationen, das Aufsplitten von Asteroiden in kleinere Asteroiden, eine Karte mit Planeten etc., Planeten mit Gravitation, andere Raumschiffe, aufsammelbare Gegenstände, PvP, ein Lebensbalken für Objekte und aufrüstbare Raumschiffe.

2.2 Retrospektive des Prozesses / Fazit

Der erste Fehler, den ich gemacht habe, war nicht gut genug nachzuforschen, was alles gefordert wird für das Projekt. Wenn ich das Projekt noch einmal machen würde, würde ich noch vor der Projektplanung gründlich zusammentragen, was genau die Anforderungen sind und welche Ressourcen zur Verfügung stehen. Die Infos für die Dokumentation habe ich beispielsweise völlig übersehen.
Was ich technisch anders machen würde, habe ich bereits in Punkt 1.5 beschrieben.
Wobei das immer erst möglich ist, wenn man einen Aha-Effekt hatte.

Zuletzt hätte ich das Projekt wenn möglich gerne mit anderen gemeinsam durchgeführt. Wobei mir bei diesem Projekt nicht die Wahl geblieben ist. Die Menge an Arbeit um die Projektarbeit herum, ging weit über das hinaus, was als Einzelperson in dem gegebenen Zeitrahmen umsetzbar ist.

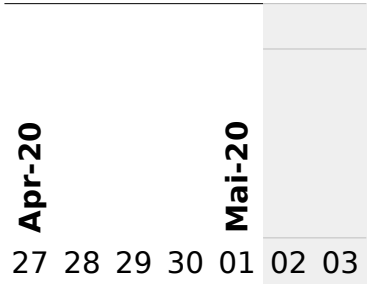
Asteroids

Third Element Aviation GmbH

Birger Möllering

Sonntag, 26. April 2020

Task Names	Resource Names	Start	Duration	Finish
		27.04.20	7	27.04.20
Ressourcen Evaluation		**27.04.20**	**1**	**27.04.20**
Ressourcen auflisten		**27.04.20**	**0,5**	**27.04.20**
Grafiken		27.04.20	0,25	27.04.20
Bibliotheken		27.04.20	0,25	27.04.20
Werkzeuge auflisten		**27.04.20**	**0,5**	**27.04.20**
Planungswerkzeuge		27.04.20	0,25	27.04.20
Implementationswerkzeuge		27.04.20	0,25	27.04.20
Milestone 1	**Ressourcenplan**	**27.04.20**	**1**	**27.04.20**
Programmplanung		**28.04.20**	**1**	**28.04.20**
Komponentenrelationen visualisieren		28.04.20	1	28.04.20
Milestone 2	**Diagramm**	**28.04.20**	**1**	**28.04.20**
Implementation		**29.04.20**	**2**	**30.04.20**
Komponenten implementieren		29.04.20	1	29.04.20
Komponenten verknüpfen		30.04.20	1	30.04.20
Milestone 3	**Asteroids**	**30.04.20**	**2**	**30.04.20**
Präsentation		**01.05.20**	**1**	**01.05.20**
Datum vereinbaren		01.05.20	0,5	01.05.20
Powerpoint slides erstellen o.ä.		01.05.20	0,5	01.05.20
Milestone 3	**Präsentation**			



Ressourcenplan Asteroids

Benötigte Grafiken

- Raumschiff ohne Feuerschweif
- Raumschiff mit Feuerschweif
- Asteroid
- Optional Gummiball

Benötige Bibliotheken

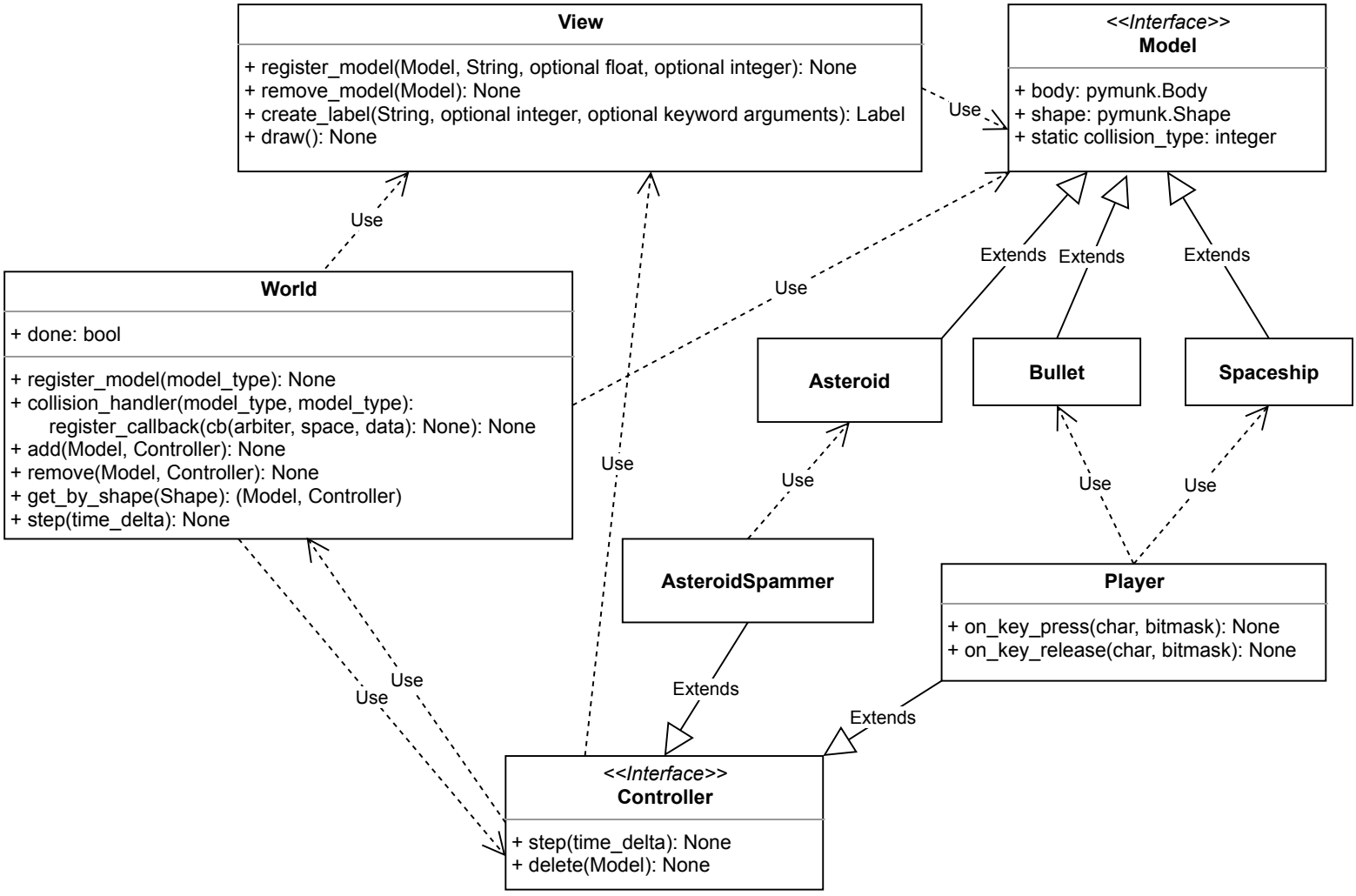
- pyglet – Cross-Platform Fenster und Multimedia Bibliothek für Python
- pymunk – 2D Physik Bibliothek für Python

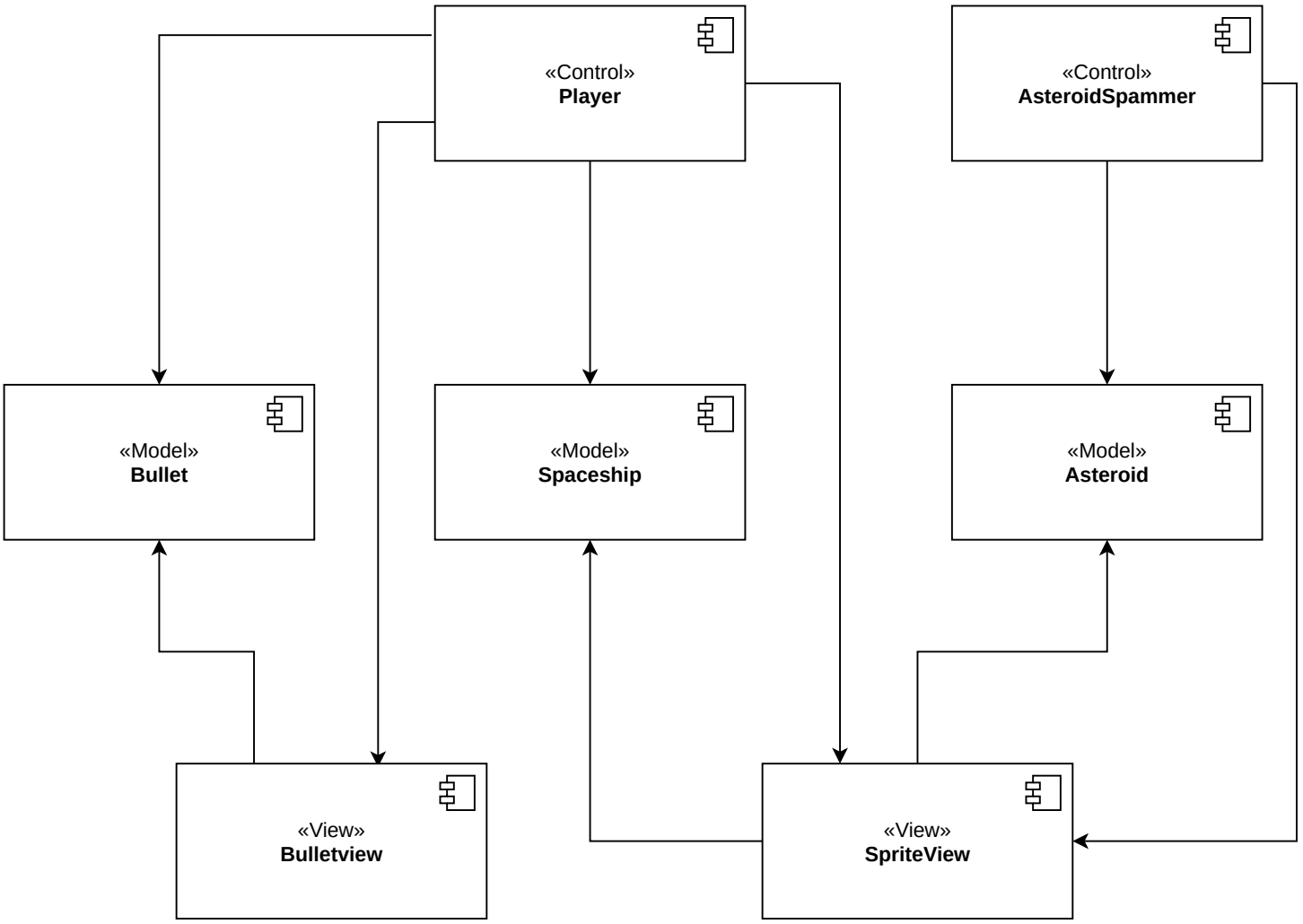
Benötigte Planungswerkzeuge

- drawio

Benötigte Implementationswerkzeuge

- git
- vim
- tmux
- virtualenv


View
+ register_model(Model, String, optional float, optional integer): None
+ remove_model(Model): None
+ create_label(String, optional integer, optional keyword arguments): Label
+ draw(): None
<<Interface>>
Model
+ body: pymunk.Body
+ shape: pymunk.Shape
+ static collision_type: integer
Use
World
+ done: bool
+ register_model(model_type): None
+ collision_handler(model_type, model_type):
register_callback(cb(arbiter, space, data): None): None
+ add(Model, Controller): None
+ remove(Model, Controller): None
+ get_by_shape(Shape): (Model, Controller)
+ step(time_delta): None
Extends
Asteroid
Bullet
Spaceship
AsteroidSpammer
Player
+ on_key_press(char, bitmask): None
+ on_key_release(char, bitmask): None
<<Interface>>
Controller
+ step(time_delta): None
+ delete(Model): None


«Control»
Player
«Control»
AsteroidSpammer
«Model»
Bullet
«Model»
Spaceship
«Model»
Asteroid
«View»
Bulletview
«View»
SpriteView