Import & Initialisierung

```
import pygame
pygame.init()
```

Dieser Code importiert das Pygame-Modul. Mit der Init-Funktion könnt ihr Pygame initialisieren, sodass Sie mit der Spielentwicklung loslegen könnt.

Fenster erzeugen

```
screen = pygame.display.set_mode((width, height))
```

Erstellt ein Fenster für dein Spiel, dass einer Leinwand ähnelt und eine Oberfläche zurückgibt. Die Oberfläche ist in einer Variable screen gespeichert. Die Argumente sind Breite (width) und Höhe (heigth) des Bildschirms.

Fenstername festlegen

```
displa y.s et_ cap tio n(S umm erBYTE 2019!)
```

Diese Funktion legt das übergebene Argument als Titel des Fensters fest.

Hauptschleife

```
for event in pygame.event.get():
        if event.type == pygame.QUIT:
                pyg ame.quit()
```

Dies ist eine Schleife, die ständig nach Ereignissen sucht. Wird das Ereignis QUIT gelesen, wird sie verlassen weiterhin wird das Einfrieren des Spiels verhindert.

Fenster aktualisieren

```
pygame.di spl ay.u pd ate()
```

Aktualisiert das Fenster und erneuert lediglich die Oberfläche, wenn keine Argumente angegeben werden. Wenn Du allerdings die Variablen spezifizierst, werden die von Dir angegebenen Teile entsprechend neu gezeichnet.

Farbe

```
pygame.Co lor(R, G, B)
```

Erzeugt ein farbiges Objekt mit RGB als Argumente.

Oberfläche färben

```
Surfac e.f ill (color)
```

Mit dieser Funktion kannst Du den ganzen Bildschirm in einer Farbe ausfüllen. Argumente sollten RGBA-Werte (Rot, Grün, Blau, Alpha) sein.

Gruppen

Eine Gruppe bilden und füllen

```
from pygame.sprite import Sprite, Group

def Sphere (Sp rite):
...
def draw_sphere(self):
 ...
 def update (self):
...
 spheres = Group()

 new_sphere = Sphere()
sphere.add(new_bullet)
```

Durchlaufen der Elemente einer Gruppe

Die sprites()-Methode gibt alle Mitglieder einer Gruppe zurück.

```
for sphere in spheres.sprites():
 sphere.dr aw_ sph ere()
```

update() auf einer Gruppe aufrufen

Beim Aufruf von update() für eine Gruppe wird automatisch update() für jedes Mitglied der Gruppe aufgerufen.

```
sphere s.u pdate()
```

Element einer Gruppe entfernen

```
sphere s.r emo ve( bullet)
```

Pygame hat eine Group-Klasse, die das Arbeiten mit einer Gruppe ähnlicher Objekte vereinfacht. Eine Gruppe ist wie eine Liste mit einigen zusätzlichen Funktionen, die beim Erstellen von Spielen hilfreich sind.

Sponsored by **ApolloPad.com**
Everyone has a novel in them. Finish Yours!
https://apollopad.com

Sprites

Sprite erstellen

```
class Player (py gam e.s pri te.S pr ite):
      def __init __( self):
            pyg ame.sp rit e.S pri te._ _i nit __(
 self)
            self.image = pygame.Su rfa ce((50, 50)
)
```

Erzeugt ein 50x50 Pixel großes Quadrat...

```
                  self.i mag e.f ill (GREEN)
```

...und färbt es grün ein.

```
                self.rect = self.i mag e.g et_ rect
()
```

Berechent das, das Objekt umschließende, Rechteck, um die Koordinaten eines Objekts zu verfolgen.

Sprite spawnen lassen

```
all_sp rites = pygame.sp rit e.G roup()
 player = Player()
 all_sp rit es.a dd (pl ayer)
```

Erst wird eine leere Gruppe für die Sprites erstellt, dann eine Sprite Player (also das Quadrat) erstellt und dieser dann auch in die Gruppe hinzugefügt.

Sprite bewegen

```
class Player (pygam e.s pri te.S pr ite):
 ...
   def update (self):
          self.r ect.x += 5
```

Um Deinen Sprite zu bewegen, musst Du zuerst Deine Klasse Player erweitern, denn es wird nun die Funktion update(self) hinzugefügt, die Dein Quadart updatet, wenn sie aufgerufen wird. In diesem Fall erhöhst Du die x-Koordinate um 5, d.h. Dein Quadrat bewegt sich nach rechts.

Kollisionen erkennen

Kollisionen zwischen einem einzelnen Objekt und einer Gruppe

Die Funktion spritecollideany() nimmt ein Objekt und eine Gruppe und gibt True zurück, wenn das Objekt mit einem Mitglied der Gruppe überlappt.

```
if pg.spr ite.sp rit eco lli dea ny( ball, pins): pins_le
ft -= 1
```

Kollisionen zwischen zwei Gruppen

sprite.groupcollide() umfasst zwei Gruppen und zwei Booleans. Sie gibt ein Wörterbuch zurück, das Infos zu den kollidierten Mitgliedern hat. Die Booleans geben an, ob entsprechende Mitglieder einer Gruppe gelöscht werden sollen.

Kollisionen erkennen (cont)

```
collisions = pg.spr ite.gr oup col lide(
 ships, aliens, True, True)
 score += len(co lli sions) * alien_ poi nt_ value
```

Du kannst erkennen, wenn ein einzelnes Objekt mit einem Mitglied einer Gruppe kollidiert. Oder auch, wenn ein Mitglied einer Gruppe mit einem Mitglied einer anderen Gruppe kollidiert.

Ereignisse

```
pygame.ev  en t.p ost()
```

Platziert ein neues Ereigniss, welches Du in der Warteschlange spezifizierst.

```
pygame.ev  en t.E v ent()
```

Erzeugt ein neues Ereignisobjekt.

```
pygame.ev  en t.get()
```

Entnimmt der Warteschlange das nächste Ereignis.

```
pygame.ev  en t.c l ear()
```

Entfernt alle Ereignisse in der Warteschlange.

Alle Ereignisse befinden sicher immer in einer Warteschlange. Dabei ist auch die Reihenfolge dieser zu beachten.

Mit mehreren Ereignissen arbeiten

```
for event in pygame.event.get():
       if event.type == pygame.KE YDOWN:
                pri nt( "Eine Taste wurde gedrüc -
kt")
       if event.type == pygame.KEYUP:
                pri nt( "Eine Taste wurde losgel -
ass en! ")
       if event.type == pygame.K_UP:
                pri nt( "Die Pfeil- Nac h-Oben
Taste wurde gedrüc kt! ")
       if event.type == pygame.K_ DOWN:
                pri nt( "Die Pfeil- Nac h-Unten
Taste wurde gedrüc kt! ")
       if event.type == pygame.K_q:
                pri nt( "Der Buchstabe Q wurde
gedrüc kt")
       if event.type == pygame.QUIT:
                pyg ame.quit()
```

Dieser Code kann beliebig verändert werden, um auch mit anderen Event Arten / Ereignis Arten zu arbeiten.

Sponsored by **ApolloPad.com**
Everyone has a novel in them. Finish Yours!
https://apollopad.com

Beenden

pygame.quit()	Beendet das Spiel

Ausgewählte Event Arten

pygame.KE YDOWN	Eine Taste wird gedrückt
pygame.KEYUP	Eine Taste wird losgelassen
pygame.MO USE MOTION	Maus Bewegung festgestellt
pygame.K_ BAC KSPACE	Rücktaste
pygame.K_ RETURN	Enter
pygame.K_ ESCAPE	ESC - Taste
pygame.K_ SPACE	Leertaste
pygame.K_a	a
pygame.K_b	b
...	...
pygame.K_z	z
pygame.K_KP0	Numpad 0
pygame.K_KP1	Numpad 1
...	...
pygame.K_KP9	Numpad 9
pygame.K_UP	Pfeil Taste nach oben
pygame.K_DOWN	Pfeil Taste nach unten
pygame.K_LEFT	Pfeil Taste nach Links
pygame.K_ RIGHT	Pfeil Taste nach Rechts

Rechtecke

Ein rect Objekt

Wir haben bereits ein Fensterobjekt. Wir können leicht auf das Rect Objekt zugreifen, welches wir dem Fensterobjekt zuordnen.

```
screen _rect = screen.ge t_r ect()
```

Mittelpunkt des Fensters bestimmen

Rect-Objekte haben ein Attribut, das den Mittelpunkt speichert.

```
screen _center = screen _re ct.c enter
```

Rechtecke (cont)

Nützliche rect Attribute

Wenn Ihr ein rect Objekt haben, gibt es eine Reihe von nützlichen Attributen, die beim Positionieren von Objekten und dem Ermitteln relativer Positionen von Objekten hilfreich sind. (Weitere Attribute findet Ihr in der Pygame Dokumentation.)

```
screen _re ct.l eft, screen_rect.right
screen_rect.top, screen_rect.bottom
screen_rect.centerx, screen_rect.centery
screen_rect.width, screen _re ct.h eight

 screen_rect.center
screen_rect.size
```

Erstellen eines rect Objektes

Ihr könnt ein Rechteck Objekt erstellen, welches eine Kugel in einem Spiel darstellen soll. Die Rect() Klasse nimmt die Koordinaten der oberen linken Ecke an. Die Werte für Breite und Höhe des Rechtecks gebt ihr als Argumente mit. Die Funktion draw.rect () benötigt zum eines das Fenster Objekt, zum zweiten die Farbe und zu guter Letzt das Rechteck Objekt.

```
bullet _rect = pygame.Re ct(100, 100, 3, 15)
color = (100, 100, 100)
pygame.draw.rect(screen, color, bullet _rect)
```

Viele Objekte in einem Spiel können als einfache Rechtecke behandelt werden. Das vereinfacht den Code, ohne das Spiel merklich zu beeinflussen. Pygame hat eine „Rect“-Objekt, das das Arbeiten mit Spielobjekten erleichtert

Sponsored by **ApolloPad.com**
Everyone has a novel in them. Finish Yours!
https://apollopad.com