Cheatography

Pygame Cheat Sheet

by ProjectSB19 via cheatography.com/81531/cs/19525/

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

```
display.set_caption(SummerBYTE 2019!)
```

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

Hauptschleife

```
for event in pygame.event.get():
    if event.type == pygame.QUIT:
        pygame.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.display.update()
```

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.Color(R, G, B)
```

Erzeugt ein farbiges Objekt mit RGB als Argumente.

Oberfläche färben

```
Surface.fill(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(Sprite):
...
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.draw_sphere()
```

update() auf einer Gruppe aufrufen

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

```
spheres.update()
```

Element einer Gruppe entfernen

```
spheres.remove(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.

Cheatography

Pygame Cheat Sheet

by ProjectSB19 via cheatography.com/81531/cs/19525/

Sprites

Sprite erstellen

```
class Player(pygame.sprite.Sprite):
   def __init__(self):
      pygame.sprite.Sprite.__init__(self)
      self.image = pygame.Surface((50, 50))
```

Erzeugt ein 50x50 Pixel großes Quadrat...

```
        self.image.fill(GREEN)
```

...und färbt es grün ein.

```
        self.rect = self.image.get_rect()
```

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

Sprite spawnen lassen

```
all_sprites = pygame.sprite.Group()
player = Player()
all_sprites.add(player)
```

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 (pygame.sprite.Sprite):
...
   def update(self):
      self.rect.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.sprite.spritecollideany(ball, pins):
pins_left -= 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.sprite.groupcollide(
ships, aliens, True, True)
score += len(collisions) * alien_point_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.event.post()
```

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

```
pygame.event.Event()
```

Erzeugt ein neues Ereignisobjekt.

```
pygame.event.get()
```

Entnimmt der Warteschlange das nächste Ereignis.

```
pygame.event.clear()
```

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.KEYDOWN:
        print("Eine Taste wurde gedrückt")
    if event.type == pygame.KEYUP:
        print("Eine Taste wurde losgelassen!")
    if event.type == pygame.K_UP:
        print("Die Pfeil-Nach-Oben Taste wurde
gedrückt!")
    if event.type == pygame.K_DOWN:
        print("Die Pfeil-Nach-Unten Taste wurde
gedrückt!")
    if event.type == pygame.K_q:
        print("Der Buchstabe Q wurde gedrückt")
    if event.type == pygame.QUIT:
        pygame.quit()
```

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

Cheatography

Pygame Cheat Sheet

by ProjectSB19 via cheatography.com/81531/cs/19525/

Beenden

`pygame.quit()`	Beendet das Spiel

Ausgewählte Event Arten

`pygame.KEYDOWN`	Eine Taste wird gedrückt
`pygame.KEYUP`	Eine Taste wird losgelassen
`pygame.MOUSEMOTION`	Maus Bewegung festgestellt
`pygame.K_BACKSPACE`	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.get_rect()
```

Mittelpunkt des Fensters bestimmen

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

```
screen_center = screen_rect.center
```

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_rect.left, screen_rect.right
screen_rect.top, screen_rect.bottom
screen_rect.centerx, screen_rect.centery
screen_rect.width, screen_rect.height

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.Rect(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