Inhaltsverzeichnis

Der Schimpanse **2**
Anforderungen **2**
Media - Dateien **2**
Vorgehen **2**
Pygame / Pygame Tools Grundlagen **3**
Das GameObject 3
Gruppen 3
Das Positionsrechteck (Rect) 3
Kollisionen 4
Richtungsangaben mit Direction 4

Der Schimpanse

Das ist eine freie Interpretation des Chimp - Beispiels auf der Webseite von Pygame.
https://www.pygame.org/docs/tut/ChimpLineByLine.html

Anforderungen

- Ein farbiger Hintergrund mit einem eingemitteten Label 'Pummel The Chimp, And Win $$$'.
- Ein Schimpanse der horizontal über das Spielfeld läuft.
- Bei jedem Richtungswechsel ändert er auch seine Blickrichtung.
- Eine Faust führt bei Druck auf eine Maustaste einen Schlag aus.
- Falls der Schlag trifft, rotiert der Schimpanse einmal um seinen Mittelpunkt. Dabei ertönt ein Geräusch.
- Falls der Schlag nicht trifft, ertönt ein anderes Geräusch.

Media - Dateien

https://github.com/pygame/pygame/tree/main/examples/data

chimp.png: der Schimpanse
fist.png: die Faust
punch.wav: Geräusch bei Treffer
whiff.wav: Geräusch bei Fehlschlag.

Vorgehen

- Projekt aus Vorlage erstellen.
- Hintergrund (self.background) mit Farbe und Text (Pummel...) erstellen.
- Spielerklasse zu Fist umbenennen und Faust darstellen.

- Faust schlägt mit Maustaste oder Leertaste.
- Gegner - Klasse zu Chimp umbenennen und über das Spielfeld laufen lassen.
- Chimp muss Blickrichtung wechseln.

- Auf Kollision der Faust mit einem Schimpansen testen.
- Bei Treffer Rotation

- Kollision durch korrigierte Hitbox anpassen. Dabei hilft ***show_boxes()***.
- Sound einbauen
- AUFGABE: Es sollen 3 Schimpansen gleichzeitig unterwegs sein. Einer davon soll vom rechten Rand her kommen.

Pygame / Pygame Tools Grundlagen

Das GameObject
Spielobjekte stellen sich automatisch dar und können sich automatisch bewegen.

Jedes Spielobjekt hat eine Eigenschaft ***image***, die das darzustellende Bild enthält. Die Position wird durch die Eigenschaft ***rect*** festgelegt. rect enthält auch die Grösse des umschliessenden Rechtecks.

Die Eigenschaft ***pos*** legt die Position unter Berücksichtigung der Ankerposition fest.

Die Ankerposition ist standardmässig die linke obere Ecke. Das wird beim Erstellen des Objekts festgelegt. Eine spätere Änderung ist aber möglich.

obj.anchor = 'c': Die Zentrumsposition wird auf die momentane Position (self.pos) gesetzt. Das Objekt verschiebt sich dadurch. Das Zentrum ist der neue Anker.

obj.change_anchor('c'): Das Objekt wird nicht verschoben. Da der Ankerpunkt verschoben wurde, ändert sich obj.pos.

Ein Gameobjekt kann mit ***.visible*** sichtbar oder unsichtbar geschaltet werden.
Mit ***.active*** kann das Objekt aktiv geschaltet werden.

Mit ***.speed*** (float) und ***.direction*** (Direction) wird sich das Objekt mit der angegebenen Geschwindigkeit (in pixel pro Sekunde) in die entsprechende Richtung bewegen.

Gruppen
Spielobjekte können Gruppen zugeordnet werden.
Zwei Standardgruppen stehen automatisch zur Verfügung: ***visible_sprites*** und ***player_sprites***.

Sichtbare Objekte gehören automatisch zur Gruppe visible_sprites. Diese werden dann automatisch dargestellt. Mit visible = False werden sie aus der Gruppe entfernt, mit visible = True wieder eingefügt.

Spieler sollten immer zur Gruppe player_sprites gehören. Das ist in der Vorlage player.py so programmiert, kann aber verändert werden.

Eigene Gruppen können einfach erstellt werden. So werden oft alle Gegner in einer Gruppe zusammengefasst.

```
self.alien_gruppe = sprite.Group()
```

Das Positionsrechteck (Rect)
https://www.pygame.org/docs/ref/rect.html

Jedes Objekt hat eine Abmessung und eine Position. Dadurch wird festgelegt, wo das Objekt dargestellt wird. Jedes Objekt kennt alle 9 Positionspunkte. Der mit ***anchor*** festgelegte Punkt ist die Ankerposition ***pos***. Standard ist oben links (anchor = 'tl').

Bei Erzeugen des Objekts kann der Ankerpunkt (Bsp.: anchor = 'br') mitgegeben werden, Standard ist 'tl'. ***obj.pos*** bezieht sich dann automatisch auf den entsprechenden Punkt.

Jedes Objekt hat ein solches Positionsrechteck. Es steht mit ***obj.rect*** oder ***obj.get_rect()*** zur Verfügung. Das Rechteck ist eigentlich Bestandteil des dargestellten Bildes (obj.image). Falls dieses ändert, muss es neu berechnet werden. ***obj.rect = obj.image.get_rect()*** stellt das neue Rechteck zur Verfügung. Dabei geht aber die Position verloren.

Die Position kann durch Ändern eines beliebigen Positionspunktes verändert werden. Alle anderen Punkte und die Darstellung passen sich automatisch an.

pos = rect.topleft, rect.bottomright += 10, rect.center ...

Kollisionen
Oft müssen Kollisionen zwischen einem Spieler und einer Gruppe Gegner festgestellt werden.

```
kollisionen = self.player.collide_with_group(self.alien_gruppe)
```

kollisionen enthält danach alle Aliens, die den Spieler berühren.

Kollisionen erfolgen normalerweise auf der Basis ihrer umhüllenden Rechtecke. Diese sind aber oft zu gross. Deshalb ist es möglich, mit ***obj.shrink_box()*** den Bereich zu verkleinern.

```
obj.shrink_box(left=0, right=0, top=0, bottom=0)
```

Zur Kontrolle können die Boxen auch angezeigt werden.

```
game.show_boxes(*groups)
```

Dabei kann eine beliebige Liste von Gruppen angegeben werden.

```
self.show_boxes(self.player_sprites, self.chimp_gruppe)
```

Das ursprüngliche Positionsrechteck wird blau angezeigt, die korrigierte Box rot.

Richtungsangaben mit Direction
Mit einem Objekt der Klasse pgt.Direction kann eine Richtung angegeben werden.

Es handelt sich dabei um einen Vektor mit der Länge 1. Eine Ausnahme ist der Vektor (0, 0), dieser hat immer die Länge 0.

`dir = Direction()`
erzeugt einen Vektor mit der Länge 0. Das wird als ***keine Richtung*** interpretiert.
Das kann auch später mit `dir.clear()` erreicht werden.

```
dir = Direction(1, 0)
dir = Direction(deg=0)
```
erzeugen Vektoren mit Länge 1, die nach rechts zeigen.

`dir = Direction(deg=180)`
erzeugt einen Vektor mit Länge 1, der nach links zeigt.
Der Winkel beginnt rechts bei 0 Grad und wird im Uhrzeigersinn gemessen.

`dir = Direction(rad=3.14159/2)`
erzeugt einen Vektor mit Länge 1, der nach unten zeigt.

Ein lesender Zugriff über die Properties `dir.x, dir.y, dir.deg` und `dir.rad` ist ebenfalls möglich.
Mit `dir.deg=` und `dir.rad=` kann der Vektor auch verändert werden.
Zuweisungen an `dir.x` oder `dir.y` sollten vermieden werden, da dabei keine Normalisierung auf Länge 1 erfolgt.

In Kombination mit einer ***Geschwindigkeit*** (float) ergibt sich eine Bewegung. Diese wird einfach durch eine Multiplikation der Richtung mit der Geschwindigkeit berechnet.