Philipps Universität Marburg

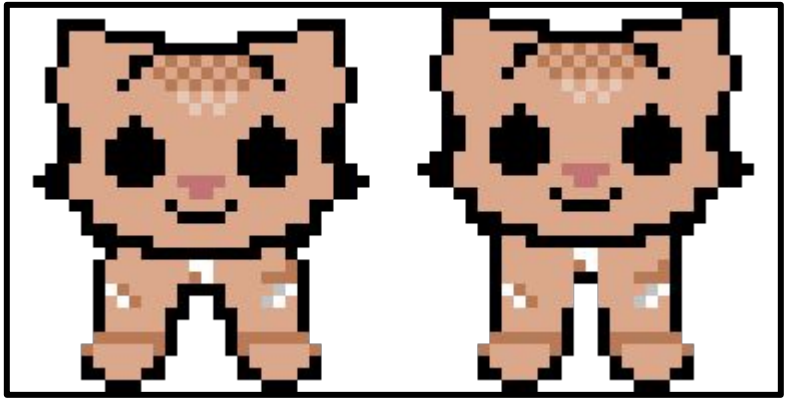
Animating Sprites

hessian.AI

Was sind Sprite Sheets?

Sprite sheets

- Auf der rechten Seite ist eine Bilddatei zu sehen.
- Die Datei stellt zwei verschiedene Sprite-Bilder dar.
- Wir bezeichnen dies als "Sprite Sheet".



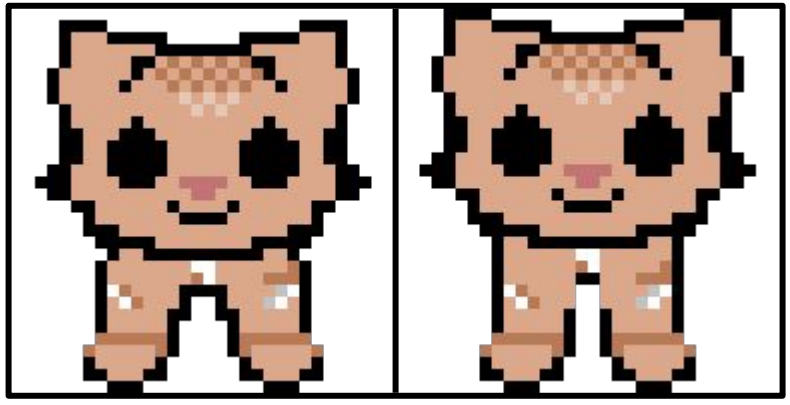
animated_sprites_tutorial/imgs/kitty-spritesheet.png

Sprite sheets: sprite images

- Wir wissen, dass das Sprite Sheet zwei durch Spalten getrennte Bilder enthält.
- Die Dimensionen der einzelnen Sprite-Bilder sind:

Width = (image file width) / #columns

Height = image file height

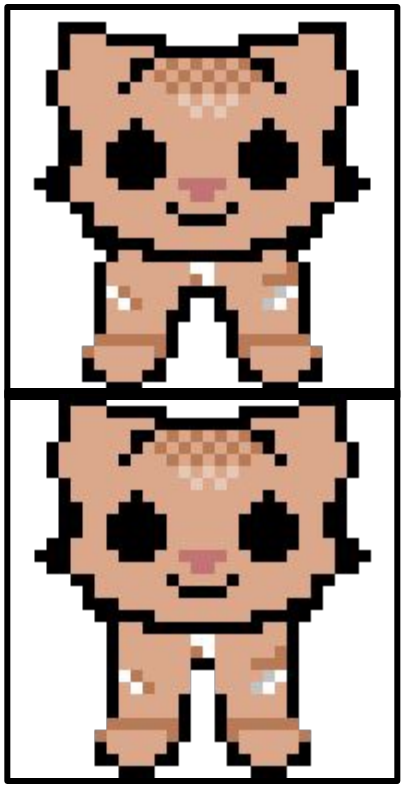


Sprite sheets: sprite images

- Alternativ können sie auch durch Zeilen getrennt werden
- Die Dimensionen der einzelnen Sprite-Bilder sind:

Width = image file width

Height = (image file height) / (#rows)

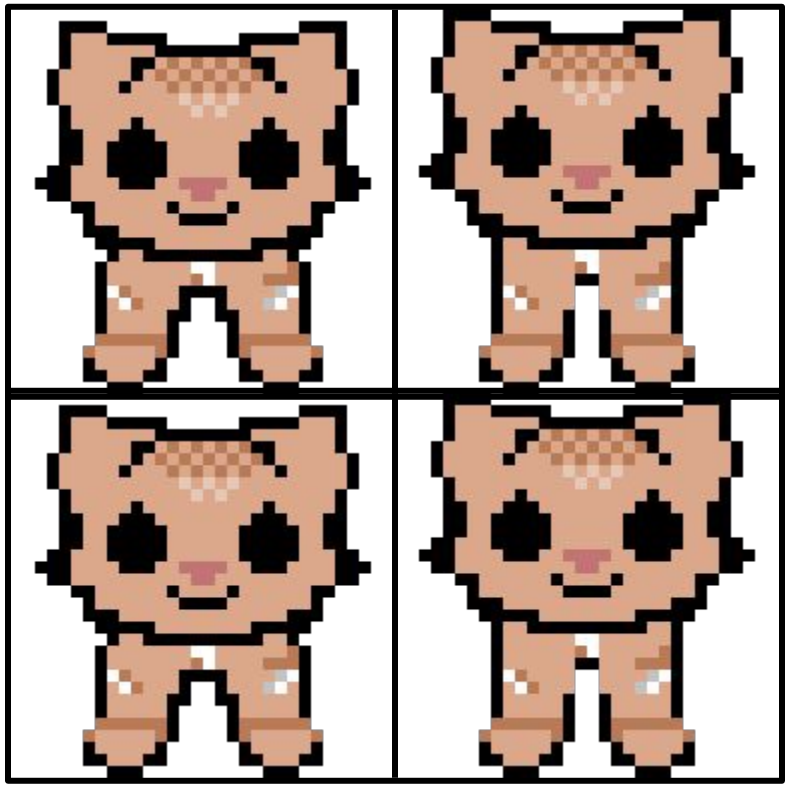


Sprite sheets: sprite images

- Es kann auch Zeilen und Spalten geben
- Die Dimensionen der einzelnen Sprite-Bilder sind:

Width = (image file width) / (#columns)

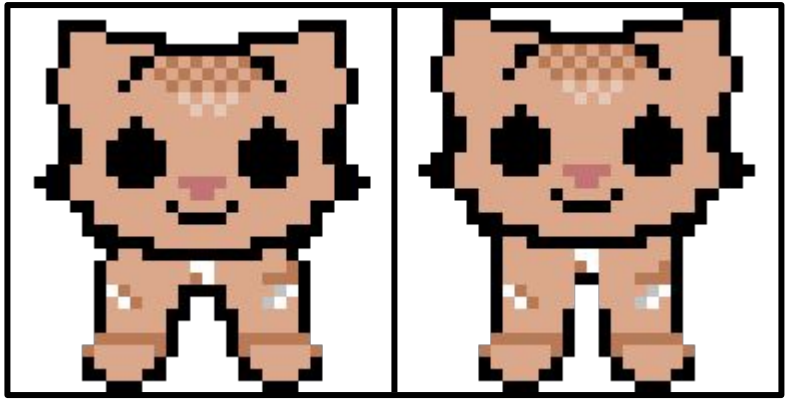
Height = (image file height) / (#rows)



Wie können wir nur ein Sprite-Bild eines Sprite-Sheets erhalten?

Unser sprite sheet

- Das ist unser Sprite Sheet
- Der Name der Datei ist "kitty-spritesheet.png".



Surfaces

- Ein Bild ist ein pygame.Surface Objekt

```
""" In tutorial_notebook.ipynb """

import pygame

spritesheet = pygame.image.load("imgs/kitty-spritesheet.png")
```

- Wir können das Rechteck der Oberfläche ermitteln

```
import pygame

spritesheet = pygame.image.load("imgs/kitty-spritesheet.png")
surface_rect = spritesheet.get_rect()
print(surface_rect) # prints <rect(0, 0, 64, 32)>
```

Surface Rects

- Wir können die Dimensionen jedes Sprite-Bildes mit dem, was wir über pygame.Rect-Objekte und das Sprite-Sheet wissen, bestimmen

```
import pygame
surface = pygame.image.load("imgs/kitty-spritesheet.png")
surface_rect = surface.get_rect()
sprite_width = surface_rect.width / 2
sprite_rect = pygame.Rect(0, 0, sprite_width, surface_rect.height)
print(surface_rect) # prints <rect(0, 0, 64, 32)>
print(sprite_rect) # prints <rect(0, 0, 32, 32)>
```

Subsurfaces

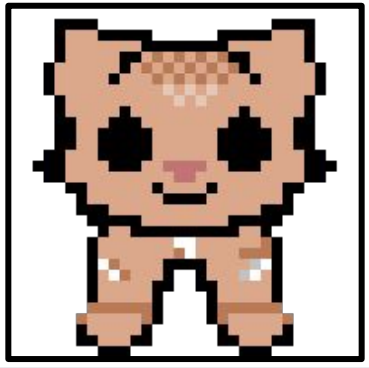
- Wir können eine Teilfläche ermitteln, indem wir ein pygame.Rect erstellen, das die Position der Teilfläche relativ zur größeren Fläche sowie die Breite und Höhe der Teilfläche angibt
- Das erste Bild beginnt bei (x,y) = (0,0) mit der Höhe und Breite, die wir gerade berechnet haben.
- Dann verwenden wir die Methode Surface.subsurface() und geben dieses Rect ein.

```
""" continues from previous code """
first_image_rect = sprite_rect
first_image = spritesheet.subsurface(first_image_rect)
```

Subsurfaces

- Wenn wir das Bild speichern, sehen wir folgendes

```
""" continues from previous code """
first_image_rect = sprite_rect
first_image = spritesheet.subsurface(first_image_rect)
pygame.image.save(first_image, "imgs/first_image.png")
```

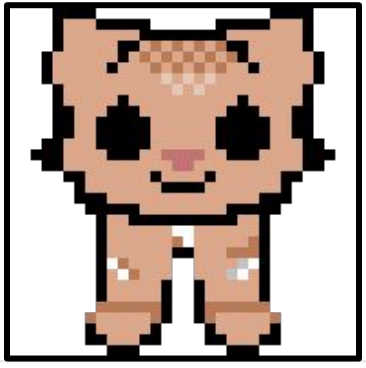


imgs/first_image.png

Subsurfaces

- Das zweite Bild beginnt an dem Pixel nach dem Ende des ersten Bildes

```
""" continues from previous code """
second_image_rect = pygame.Rect(sprite_width, 0, sprite_width, surface_rect.height)
second_image = spritesheet.subsurface(second_image_rect)
pygame.image.save(second_image, "imgs/second_image.png")
```



imgs/second_image.png

Ändern des Sprite Images in der Game Loop

0-immersive-kitty.py

- Wir betrachten die Datei “0-immersive-kitty.py”
- Gehen wir die einzelnen Teile des Codes durch

Imports, initialization, constants

- Das alles ist euch bekannt

```
import pygame, sys
from pygame.locals import *
import utils

pygame.init()
pygame.display.set_caption('Sprite sheets')
icon = utils.load_image("imgs/kitty.png", size=(32,32))
pygame.display.set_icon(icon)
SCREENRECT = pygame.Rect(0, 0, 640, 480)
WHITE = (225,225,225)
```

Funktion: load sprite sheet

- Hier haben wir eine Funktion, die das Sprite Sheet lädt, so dass jede Teilfläche ein Element in der Liste 'images' ist

```
def load_spritesheet(file_path, cols=1, scale=None, size=None):
    surface = utils.load_image(file_path, scale=scale, size=size)
    sheet_rect = surface.get_rect()
    image_width = int(sheet_rect.width / cols)

    images = []
    for i in range(0, sheet_rect.width, image_width):
        rect = pygame.Rect(i, 0, image_width, sheet_rect.height)
        image = surface.subsurface(rect)
        images.append(image)
    return images
```

Player class

Klasse Player, die etwas anders als zuvor aussieht

```
class Player(pygame.sprite.Sprite):
    speed = 10
    def __init__(self, sprite_sheet) -> None:
        super().__init__()
        self.sprite_sheet = sprite_sheet
        self.current_image = 0
        self.image = self.sprite_sheet[self.current_image]
        self.rect = self.image.get_rect()

    def update(self):
        self.current_image = 1 if self.current_image == 0 else 0
        self.image = self.sprite_sheet[self.current_image]

    def move(self, keystates, boundary):
        dx = self.speed * (keystates[K_RIGHT] - keystates[K_LEFT])
        self.rect.move_ip(dx, 0)
        self.rect.clamp_ip(boundary)
```

Game loop in main function

- Die teile der main function und der Game Loop sind euch bekannt
- Bei Unklarheiten einfach fragen
- Schaut euch das ende (clock.tick) der Schleife an

```
        """
        - clock.tick(FPS) returns milliseconds since last tick
        - to get seconds, we can divide by 1000
        """
        dt = clock.tick(FPS) / 1000
        print(dt)
```

Run the code

- Die FPS beträgt 2 Frames pro Sekunde.
- Daher sollte sich das Sprite-Bild jede Sekunde vom ersten Bild zum zweiten Bild ändern.
- Wir geben **dt** aus, das heißt wir bekommen 0.5 (circa) ausgegeben

animated_sprites_tutorial/imgs/animated_cat.gif

Moving the sprite

- Wenn wir das Sprite mit den Rechts- und Linkspfeilen von links nach rechts bewegen, aktualisiert das Sprite seine Position zur gleichen Zeit wie das Bild.
- Dadurch sieht die Animation so aus, als würde die Katze von einer Seite zur anderen springen.
- Wir möchten, dass die Animation nicht mit den Positionsänderungen synchronisiert wird, damit es so aussieht, als würde die Katze nicht laufen.

Animation basiert auf der tatsächlichen Zeit, nicht auf den Frames pro Sekunde

1-immersive-kitty.py

- Seht euch die Datei "1-immersive-kitty.py" an.
- Gehen wir den Code durch, der sich von 0-immersive-kitty.py unterscheidet

Main function

- **FPS = 10** anstatt 2
- **sprites.update(dt)**: wir übergeben dt an update
  - *Bemerkung: Wenn wir mehr Sprites haben, wäre es besser, die eigene Update-Funktion des Spielers anders zu benennen, sonst müssten wir die Update-Methode der anderen Sprites überschreiben, um dt einzubeziehen.*

Player class

- **self.time_since_update = 0**: Wir haben eine Member-Variable namens time_since_update hinzugefügt, die auf 0 gesetzt wird, wenn ein Player-Objekt erstellt wird
- **Player.update method**:
  - Wir haben den Parameter ‘dt’ hinzugefügt
  - **self.time_since_update += dt**: wir erhöhen den Wert von **time_since_update** um **dt**
  - **if self.time_since_update >= 0.5**: Wenn dt 0,5 Sekunden oder mehr ist, aktualisieren wir das Sprite Image, und setzen **self.time_since_update** auf **0**

Run the code

- Jetzt aktualisiert das Sprite sein Bild und seine Position mit unterschiedlichen Zeiten

animated_sprites_tutorial/imgs/update-sprite-image.gif

Baut das Gelernte in euer eigenes Spiel ein

Ideen

Der nächste Teil liegt in eurer Hand! Hier sind einige Ideen, wie ihr das Gelernte anwenden könnt.

Animiere die Cat im cat wants peppers game

Füge die Cat Animation in das Pepper game (vom Anfang der Woche) ein

Du musst dir überlegen, wie du die Aktualisierungen der Spieler- und Pepper-Sprites anders handhaben willst. Ist es sinnvoll, sie in verschiedene Sprite-Gruppen zu stecken? Oder Player.update anders zu benennen und es separat in der Game Loop aufzurufen?

Es gibt viele mögliche Optionen, die ihr in eurem Code anwenden könnt. Versucht, einen Code zu erstellen, der für euch logisch erscheint. Diskutiert untereinander über andere mögliche Ansätze und nehmt alle Änderungen vor, von denen ihr glaubt, dass sie den Code verbessern würden.

Animiere die Cat wenn sie die Pepper fängt

Die Cat könnte sich kurz strecken, wenn sie die Pepper fängt. Überlegt euch ein *bedingtes* update des player's sprite image.

Es gibt viele mögliche Optionen, die ihr in eurem Code anwenden könnt. Versucht, einen Code zu erstellen, der für euch logisch erscheint. Diskutiert untereinander über andere mögliche Ansätze und nehmt alle Änderungen vor, von denen ihr glaubt, dass sie den Code verbessern würden.

Entwerft eure eigenen animierten Spiele!

- Habt ihr schon Ideen? Dann probiert sie aus!
- Denkt an die drei Teile der Game Loop. Es ist hilfreich zu notieren, welches Ereignis die Update-Funktion auslöst, was in der Update-Funktion aktualisiert werden muss und wann (relativ zur main Game Loop) und wo genau das Objekt auf dem Bildschirm gezeichnet werden soll.
- Diskutiert untereinander über die Logik und den Plan für euren Code.
- Ihr möchtet euren eigenen Sprite Sheets entwerfen? Kein Problem. Diese müsst ihr nur ebenfalls in euren Code einbauen, damit ihr sie vorführen könnt. Nutzt dazu online tools wie https://www.pixilart.com/ oder https://www.piskelapp.com/ um eure Sprites zu entwerfen.