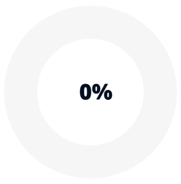
Plagiarism report

utils.py

Scan details

Scan time:	Total Pages:	Total Words:
April 13th, 2024 at 9:29 UTC	3	700

Plagiarism Detection



Types of plagiarism		Words
● Identical	0%	0
● Minor Changes	0%	0
● Paraphrased	0%	0
○ Omitted Words	0%	0

AI Content Detection

Text coverage

- ● AI text
- ○ Human text

Plagiarism Results: No results found!

```python
"""
This module contains utility functions and classes for loading and managing game assets such as images and anima
It provides functionality to load individual images or sequences of images, and to animate sequences of images w
control over animation speed and looping behavior.
"""

import os
import pygame

BASE_IMG_PATH = 'data/pngs/'

def load_image(path):
    """
    Load an image from the specified path, setting a color key for transparency.

    Parameters:
        path (str): The path relative to the base image directory to load the image from.

    Returns:
        pygame.Surface: The loaded image with a transparent background set to black (0, 0, 0).
    """
    img = pygame.image.load(BASE_IMG_PATH + path).convert()
    img.set_colorkey((0, 0, 0))
    return img

def load_pngs(path):
    """
    Load all PNG images from a specified directory path, sorted by file name.

    Parameters:
        path (str): The directory path relative to the base image directory containing PNG images to load.

    Returns:
        list[pygame.Surface]: A list of loaded pygame.Surface objects for each PNG image in the directory.
    """
    pngs = []
    for img_name in sorted(os.listdir(BASE_IMG_PATH + path)):
        pngs.append(load_image(path + '/' + img_name))
    return pngs

class Animation:
    """
    A class for managing animations by cycling through a sequence of images.

    Attributes:
        pngs (list[pygame.Surface]): A list of images for each frame of the animation.
        loop (bool): Whether the animation should loop.
        img_duration (int): How many updates each image in the sequence should be displayed for.
        done (bool): Whether the animation has finished (relevant for non-looping animations).
        frame (int): The current frame of the animation.
    """

    def __init__(self, pngs, img_dur=5, loop=True):
        """
        Initializes a new Animation instance.

        Parameters:
            pngs (list[pygame.Surface]): A list of pygame.Surface objects for each frame of the animation.
            img_dur (int, optional): Duration (in update cycles) to display each frame. Defaults to 5.
            loop (bool, optional): Whether the animation should loop indefinitely. Defaults to True.
        """
        self.pngs = pngs
        self.loop = loop
        self.img_duration = img_dur
        self.done = False
        self.frame = 0

    def copy(self):
        """
        Creates a copy of this Animation instance.

        Returns:
            Animation: A new Animation instance with the same images, duration, and loop setting.
        """
        return Animation(self.pngs, self.img_duration, self.loop)

    def update(self):
        """
        Updates the animation's current frame, progressing the animation.

        This method should be called every update cycle. It advances the frame count, and handles looping
        or marking the animation as done if it does not loop.
        """
        if self.loop:
            self.frame = (self.frame + 1) % (self.img_duration * len(self.pngs))
        else:
            self.frame = min(self.frame + 1, self.img_duration * len(self.pngs) - 1)
            if self.frame >= self.img_duration * len(self.pngs) - 1:
                self.done = True

    def img(self):
        """
        Retrieves the current image of the animation based on the current frame.

        Returns:
            pygame.Surface: The current frame's image.
        """
        return self.pngs[int(self.frame / self.img_duration)]
```