Copyleaks

Plagiarism report

tilemap.py

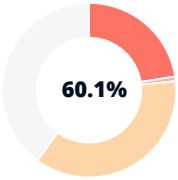
Scan details

Scan time:
April 13th, 2024 at 9:29 UTC

Total Pages:
7

Total Words:
1743

Plagiarism Detection



Types of plagiarism		Words
Identical	22.5%	392
Minor Changes	0.9%	15
Paraphrased	36.7%	640
Omitted Words	0%	0

AI Content Detection

Text coverage
- AI text
- Human text

Plagiarism Results: (10)

Pylance crashing for unknown reason · Issue #4712 · microsoft/pylance-r... 31.2%

https://github.com/microsoft/pylance-release/issues/4712

Skip to content ...

Pygame Platformer 03 — Physics, Output data of rect, Jump | by Yen | M... 29.3%

https://medium.com/@monlesyen/pygame-platformer-03-physics-output-data-of-rect-jump-d75947dae66

Yen

Open in appSign up Sign in Write Sign up Sign in Pygame Platformer 03 — Physics, Output data of rect, Jump ...

python - Pygame Platformer Collisions - Stack Overflow 25.6%

https://stackoverflow.com/questions/78196410/pygame-platformer-collisions

Stack Overflow About ...

Reddit - Dive into anything 24.2%

https://www.reddit.com/r/pygame/comments/15ldj02/typeerror_int_object_is_not_subscriptable/

Skip to main content ...

Results

Pygame Platformer 07 — Particles. This is the note taken while I was... | ... 17%

https://medium.com/@monlesyen/pygame-platformer-07-particles-fed706d34c5d

Yen

Open in appSign up Sign in Write Sign up Sign in Pygame Platformer 07 — Particles Yen ·Follow 3 m...

Pygame Platformer 04 — Camera & Background | by Yen | Medium 13.8%

https://medium.com/@monlesyen/pygame-platformer-04-camera-background-893b61a37163

Yen

Open in appSign up Sign in Write Sign up Sign in Pygame Platformer 04 — Camera & Background Yen ·...

Pygame Platformer 06 — Level Design II | by Yen | Medium 7.4%

https://medium.com/@monlesyen/pygame-platformer-06-level-design-c916706b4e41

Yen

Open in appSign up Sign in Write Sign up Sign in Pygame Platformer 06 — Level Design II Yen ·Follow ...

Pygame Platformer 02 — Collider, Utils, Entities, Tilemap & Render | by Y... 4.4%

https://medium.com/@monlesyen/pygame-platformer-02-collider-utils-entities-tilemap-1acc19db9323

Yen

Open in appSign up Sign in Write Sign up Sign in Pygame Platformer 02 — Collider, Utils, Entities, Tilemap &...

Your File 2.5%

render(self, surf, offset=(0, 0)):

Your File 1.7%

render(self, surf, offset=(0, 0)):

```
"""
This module handles the creation, management, and rendering of tilemaps for the game. It provides functionalitie
for loading and saving tilemaps from files, accessing and modifying the tiles within the map, and rendering the
tilemap to the screen. It also includes utilities for tile-based collision detection and automatic tile variant
selection based on neighboring tiles to create more visually cohesive maps.
"""

import json
import pygame


# Offsets for determining tile neighbors.
NEIGHBOR_OFFSETS = [(-1, 0), (-1, -1), (0, -1), (1, -1), (1, 0), (0, 0), (-1, 1), (0, 1), (1, 1)]

# Set of tile types that are considered solid for collision purposes.
tiles_set = {'grass', 'stone', 'sky'}


class Tilemap:
    """
    Manages a grid of tiles for the game world, including loading, saving, rendering, and collision detection.

    Attributes:
        game (Game): The main game object, providing access to global resources and settings.
        tile_size (int): The size of each tile in pixels.
        tilemap (dict): A dictionary representing the grid of tiles, with keys as string coordinates "x;y".
        offgrid_tiles (list): A list of tiles that do not align to the grid, used for decorative elements.
    """

    def __init__(self, game, tile_size=16):
        """
        Initializes a new Tilemap instance.

        Parameters:
            game (Game): The main game object.
            tile_size (int, optional): The size of each tile in pixels. Defaults to 16.
        """
        self.game = game
        self.tile_size = tile_size
        self.tilemap = {}
        self.offgrid_tiles = []

    def extract(self, id_pairs, keep=False):
        """
        Extracts tiles from the map matching specific id and variant pairs.

        Parameters:
            id_pairs (list of tuple): A list of tuples (type, variant) to match tiles against.
            keep (bool, optional): Whether to keep the matched tiles in the map. Defaults to False.

        Returns:
            list: A list of matched tiles with their properties.
        """
        # Extrahuje dlaždice z mapy na základě zadaných kombinací identifikátorů
        matches = []  # Inicializace prázdného seznamu pro shody

        # Iterace přes dlaždice mimo mřížku
        for tile in self.offgrid_tiles.copy():
            if (tile['type'], tile['variant']) in id_pairs:
                matches.append(tile.copy())
                if not keep:
                    self.offgrid_tiles.remove(tile)

        # Iterace přes dlaždice v mřížce
        for loc in list(self.tilemap.keys()):
            tile = self.tilemap[loc]
            if (tile['type'], tile['variant']) in id_pairs:
                matches.append(tile.copy())
                matches[-1]['pos'] = matches[-1]['pos'].copy()
                matches[-1]['pos'][0] *= self.tile_size
                matches[-1]['pos'][1] *= self.tile_size
                if not keep:
                    del self.tilemap[loc]

        return matches  # Vrátí nalezené shody

    def tiles_around(self, pos):
        """
```

```
        Gets a list of tiles surrounding a specific position.

        Parameters:
            pos (tuple): The position to check around, in pixels.

        Returns:
            list: A list of tiles around the specified position.
        """
        # Vrací seznam dlaždic kolem zadané pozice
        tiles = []  # Inicializace seznamu pro dlaždice
        tile_loc = (int(pos[0] // self.tile_size), int(pos[1] // self.tile_size))  # Určení umístění dlaždice
        # Iterace přes posuny sousedních dlaždic
        for offset in NEIGHBOR_OFFSETS:
            check_loc = str(tile_loc[0] + offset[0]) + ';' + str(tile_loc[1] + offset[1])
            if check_loc in self.tilemap:
                tiles.append(self.tilemap[check_loc])
        return tiles  # Vrátí seznam sousedních dlaždic

    def save(self, path):
        """
        Saves the current state of the tilemap to a file.

        Parameters:
            path (str): The file path to save the tilemap to.
        """
        # Uloží stav dlaždic do souboru ve formátu JSON
        with open(path, 'w') as f:
            json.dump({'tilemap': self.tilemap, 'tile_size': self.tile_size, 'offgrid': self.offgrid_tiles}, f)

    def load(self, path):
        """
        Loads a tilemap from a file, replacing the current state.

        Parameters:
        path (str): The file path to load the tilemap from.
        """
        # Načte stav dlaždic ze souboru ve formátu JSON
        with open(path, 'r') as f:
            map_data = json.load(f)

        # Limit specific values in the map_data to not exceed 2
        # Example for 'level', 'entity_count', or similar keys:
        if 'level' in map_data:
            map_data['level'] = min(map_data['level'], 2)
        if 'entity_count' in map_data:
            map_data['entity_count'] = min(map_data['entity_count'], 2)
        # Add similar checks for other keys as needed

        self.tilemap = map_data['tilemap']
        self.tile_size = map_data['tile_size']
        self.offgrid_tiles = map_data['offgrid']

    def solid_check(self, pos):
        """
        Checks if a given position collides with a solid tile.

        Parameters:
            pos (tuple): The position to check, in pixels.

        Returns:
            bool: True if there is a collision with a solid tile, False otherwise.
        """
        # Zkontroluje, zda je na dané pozici pevná dlaždice
        tile_loc = str(int(pos[0] // self.tile_size)) + ';' + str(int(pos[1] // self.tile_size))
        if tile_loc in self.tilemap:
            if self.tilemap[tile_loc]['type'] in tiles_set:
                return self.tilemap[tile_loc]  # Vrátí dlaždici

    def physics_rects_around(self, pos):
        """
        Gets a list of pygame.Rect objects for solid tiles around a specified position, for physics calculations

        Parameters:
            pos (tuple): The position to check around, in pixels.

        Returns:
            list: A list of pygame.Rect objects for solid tiles around the specified position.
        """
```

```
    # Vrací seznam obdélníků pro kolize kolem zadané pozice
    rects = []  # Inicializace seznamu obdélníků
    for tile in self.tiles_around(pos):
        if tile['type'] in tiles_set:
            rects.append(pygame.Rect(tile['pos'][0] * self.tile_size, tile['pos'][1] * self.tile_size, self.
    return rects  # Vrátí seznam obdélníků



def render(self, surf, offset=(0, 0)):
    """
    Renders the tilemap onto a given surface.

    Parameters:
        surf (pygame.Surface): The surface to render the tilemap on.
        offset (tuple): An (x, y) offset to apply to the tile positions, used for camera movement.
    """
    # Vykreslí dlaždice na zadaný povrch s dan
    # Vykreslení dlaždic mimo mřížku na herní plochu s použitím jejich pozic a variant
    for tile in self.offgrid_tiles:
        surf.blit(self.game.assets[tile['type']][tile['variant']], (tile['pos'][0] - offset[0], tile['pos'][

    # Procházení oblasti herní plochy, která je viditelná na obrazovce
    for x in range(offset[0] // self.tile_size, (offset[0] + surf.get_width()) // self.tile_size + 1):
        for y in range(offset[1] // self.tile_size, (offset[1] + surf.get_height()) // self.tile_size + 1):
            # Určení lokace dlaždice na základě souřadnic x a y
            loc = str(x) + ';' + str(y)
            # Kontrola, zda se dlaždice nachází v self.tilemap na dané lokaci
            if loc in self.tilemap:
                # Načtení dlaždice z self.tilemap
                tile = self.tilemap[loc]
                # Vykreslení dlaždice na herní plochu na příslušnou pozici podle offsetu
                surf.blit(self.game.assets[tile['type']][tile['variant']], (tile['pos'][0] * self.tile_size
```