**Ja:** Implement a `Kamera` (camera) class in `triedy.kamera`

**AI:** I'll help you implement a `Kamera` class that will handle camera movement and following the player in the game. Based on the codebase, we'll need to integrate it with the existing sprite and scene system.

Here's the implementation for `triedy/kamera.py`:

"""

Modul pre kameru.

"""

import typing as t

import pygame

from triedy.sprite.entity import Hrac

class Kamera:

"""

Kamera, ktorá sleduje hráča a posúva všetky objekty na obrazovke.

Používa sa staticky (nie je potrebné vytvárať viac ako jednu inštanciu).

"""

OFFSET = pygame.Vector2(0, 0)

"""Aktuálny posun kamery."""

@staticmethod

def sleduj\_hraca(hrac: Hrac, velkost\_okna: t.Tuple[int, int]):

"""

Nastaví pozíciu kamery tak, aby sledovala hráča.

"""

Kamera.OFFSET.x = hrac.rect.centerx - velkost\_okna[0] // 2

Kamera.OFFSET.y = hrac.rect.centery - velkost\_okna[1] // 2

@staticmethod

def aplikuj\_na\_sprite(sprite: pygame.sprite.Sprite) -> pygame.Rect:

"""

Aplikuje posun kamery na daný sprite a vráti jeho nový rect.

"""

return pygame.Rect(

sprite.rect.x - Kamera.OFFSET.x,

sprite.rect.y - Kamera.OFFSET.y,

sprite.rect.width,

sprite.rect.height,

)

Now we need to modify the `Level` class to use the camera. Here's how to update `triedy/scena/levely/level.py`:

import nastavenia as n

from triedy.kamera import Kamera

class Level(Scena):

def \_\_init\_\_(self, mapa\_id: str):

super().\_\_init\_\_()

self.nacitat\_level(mapa\_id)

self.hrac = None

def nacitat\_level(self, mapa\_id: str):

self.mapa = pytmx.load\_pygame(

str(self.LEVELY\_ROOT / f"{mapa\_id}.tmx"), pixelalpha=True

)

for layer in self.mapa.visible\_layers:

if isinstance(layer, pytmx.TiledTileLayer):

for x, y, gid, \*\_ in layer.iter\_data():

tile = self.mapa.get\_tile\_image\_by\_gid(gid)

if tile:

sprite = Sprite(

(x \* self.mapa.tilewidth, y \* self.mapa.tileheight)

)

sprite.image = tile

self.add(sprite)

elif isinstance(layer, pytmx.TiledObjectGroup):

for obj in layer:

if obj.name == "hrac":

self.hrac = Hrac((obj.x, obj.y))

self.add(self.hrac)

def update(self):

super().update()

if self.hrac:

Kamera.sleduj\_hraca(self.hrac, n.VELKOST\_OKNA)

def draw(self, surface: pygame.Surface):

for sprite in self.sprites():

surface.blit(sprite.image, Kamera.aplikuj\_na\_sprite(sprite))

The camera will keep the player centered on the screen as they move around the level. The camera offset is applied to all sprites during rendering, creating the illusion of camera movement.

To use this, no changes are needed in other files since we're using the existing sprite and scene system. The camera will automatically work when levels are rendered.

**Ja:** can you make the collision detection use masks instead?

so, when we are building the level, we also build a mask out of the wall positions and save it, then we use that to check the collisions, as it is a lot faster

**AI**: I'll help modify the code to use mask-based collision detection. We'll need to create a combined mask for all walls and use it for collision checking. Here are the necessary changes:

class Level(Scena):

def \_\_init\_\_(self, mapa\_id: str):

super().\_\_init\_\_()

# ...

self.steny\_maska = None

def nacitat\_level(self):

self.mapa = pytmx.load\_pygame(

str(self.LEVELY\_ROOT / f"{self.mapa\_id}.tmx"), pixelalpha=True

)

# ... existing podlaha loading code ...

# Create a surface for walls mask

maska\_povrch = pygame.Surface((self.mapa.width \* self.mapa.tilewidth,

self.mapa.height \* self.mapa.tileheight), pygame.SRCALPHA)

steny: pytmx.TiledTileLayer = self.mapa.get\_layer\_by\_name("steny") # type: ignore

for x, y, image in steny.tiles():

pozicia = (x \* self.mapa.tilewidth, y \* self.mapa.tileheight)

sprite = Stena(pozicia)

sprite.image = image

self.add(sprite)

maska\_povrch.blit(image, pozicia)

self.steny\_maska = pygame.mask.from\_surface(maska\_povrch)

# ... existing entity loading code ...

def skontroluj\_kolizie(self):

if not self.hrac:

return

hrac\_maska = pygame.mask.from\_surface(self.hrac.image)

offset = (int(self.hrac.rect.x), int(self.hrac.rect.y))

if self.steny\_maska.overlap(hrac\_maska, offset):

self.hrac.rect.x = self.hrac.posledna\_pozicia[0]

self.hrac.rect.y = self.hrac.posledna\_pozicia[1]

**Ja:** in `Level.draw`, make it so that `Podlaha` is always on bottom (so, it is drawn first)

**AI:** I'll modify the sorting key in the draw method to ensure Podlaha sprites are always drawn first. Here's the change:

def draw(self, surface: pygame.Surface):

tmavy\_povrch = self.tmavy\_povrch.copy()

# zoradenie spritov podľa Y pozície (hĺbky)

for sprite in sorted(

self.sprites(),

key=lambda sprite: [

not isinstance(sprite, Podlaha), # Podlaha will be False (0), others True (1)

sprite.rect.y,

],

):

surface.blit(sprite.image, Kamera.aplikuj\_na\_sprite(sprite))

if isinstance(sprite, SvetelnaEntita):

sprite.svetlo.aplikuj\_na\_tmu(tmavy\_povrch)

surface.blit(tmavy\_povrch, (0, 0), special\_flags=pygame.BLEND\_RGBA\_MIN)

The change inverts the `Podlaha` check in the sorting key and moves it to the first position.