```python
import pygame as pg
from lib.inject_arguments import inject_arguments

from mario.bridge.events.game_events import *


class FrameReader:
    """Read the Frame event and make other game events """

    @inject_arguments
    def __init__(self, event_dispatcher):
        self.event_dispatcher.listen('game.frame', self.handle_frame)
        self.frame = None

    @inject_arguments
    def handle_frame(self, frame):
        """Handle Frame event """

        self.build_events('game.block', Block,
            ['brick_group', 'coin_box_group', 'ground_group', 'pipe_group', 'step_group'], frame)
        self.build_events('game.enemy', Enemy, ['enemy_group'], frame)
        self.build_events('game.powerup', Powerup, ['powerup_group'], frame)
        self.build_events('game.coin', Coin, ['coin_group'], frame)


    def build_events(self, event_name, event_class, groups, frame):
        """Build DetectedComponent game event for each displayed sprite of the groups  """

        sprites = []
        for group in groups:
            sprites.extend(frame.sprite_groups[group].sprites())
        viewport_sprite = ViewportSprite(frame.viewport)
        displayed_sprites = pg.sprite.spritecollide(viewport_sprite, sprites, False)

        # Make the events and dispatch
        for block in displayed_sprites:
            self.event_dispatcher.dispatch(event_name, event_class(block.rect, frame.mario.rect, frame.current_frame))
            # print('game.block', Block(block.rect, frame.mario.rect, frame.current_frame))

        #debug
        pg.display.set_caption("Displayed blocks =  " + str(len(displayed_sprites)))


class ViewportSprite(pg.sprite.Sprite):
    """A false sprite containing viewport """

    def __init__(self, viewport_rect):
        pg.sprite.Sprite.__init__(self)
        self.rect = viewport_rect
```