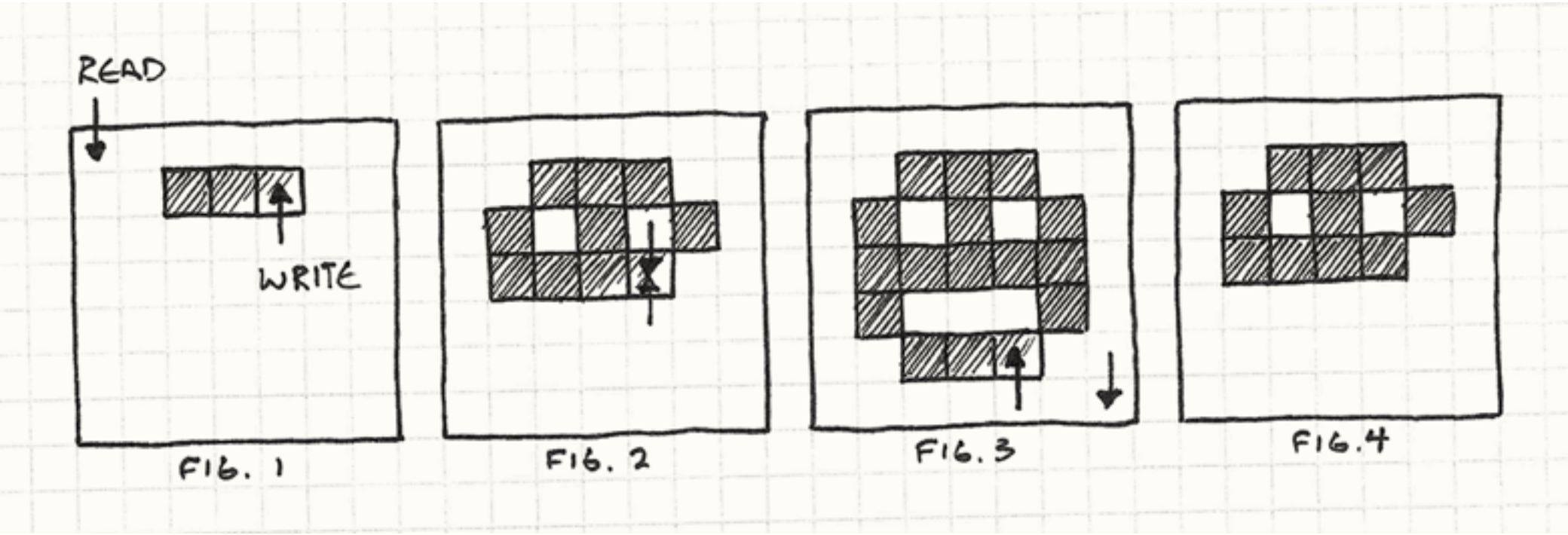
6 Images, Vector Graphics, and Scenes

Literature:

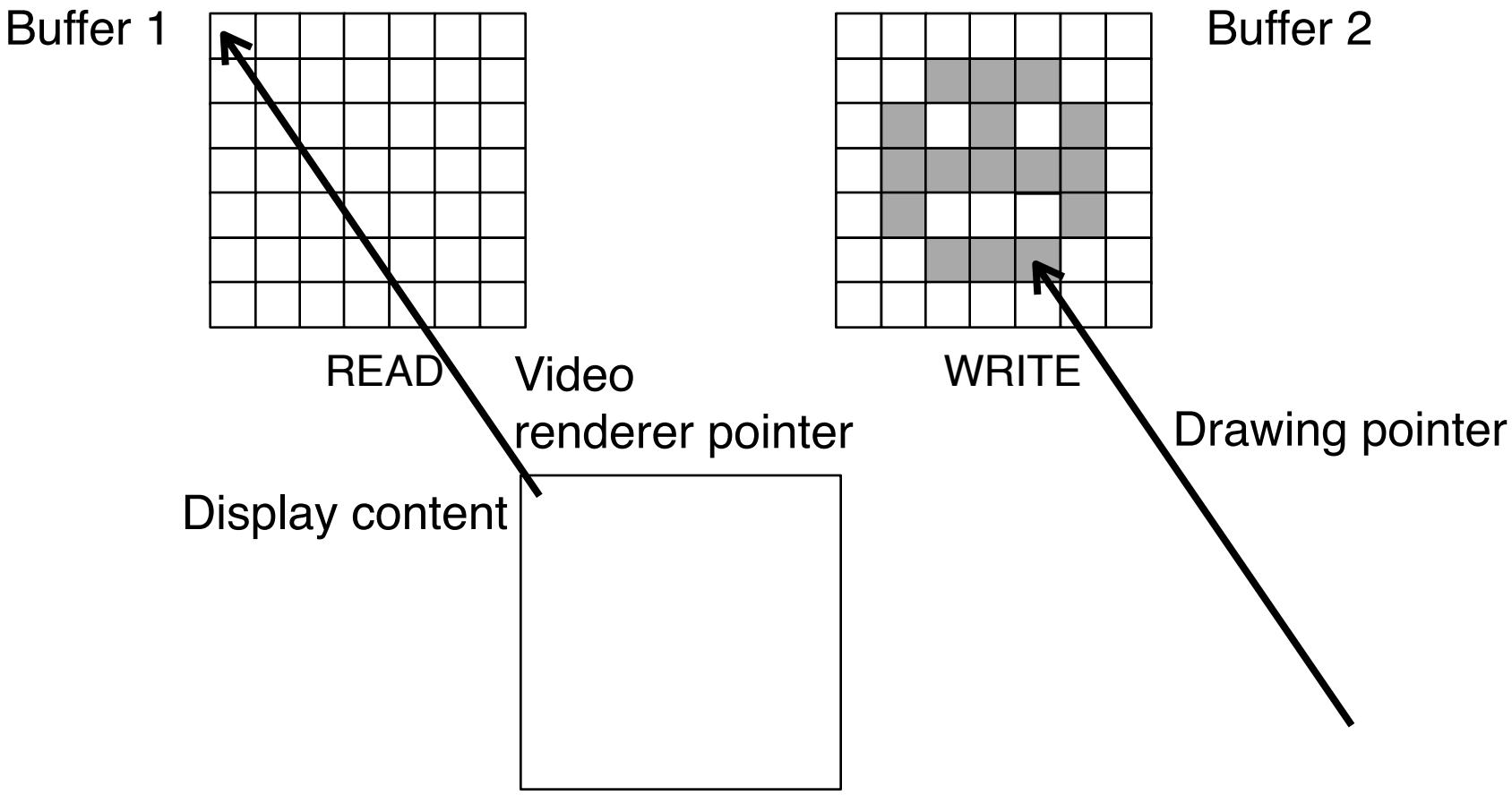
R. Nystrom: Game Programming Patterns, genever banning 2014, Chapter 8, see also http://gameprogrammingpatterns.com/double-buffer.html

Tearing and Flickering

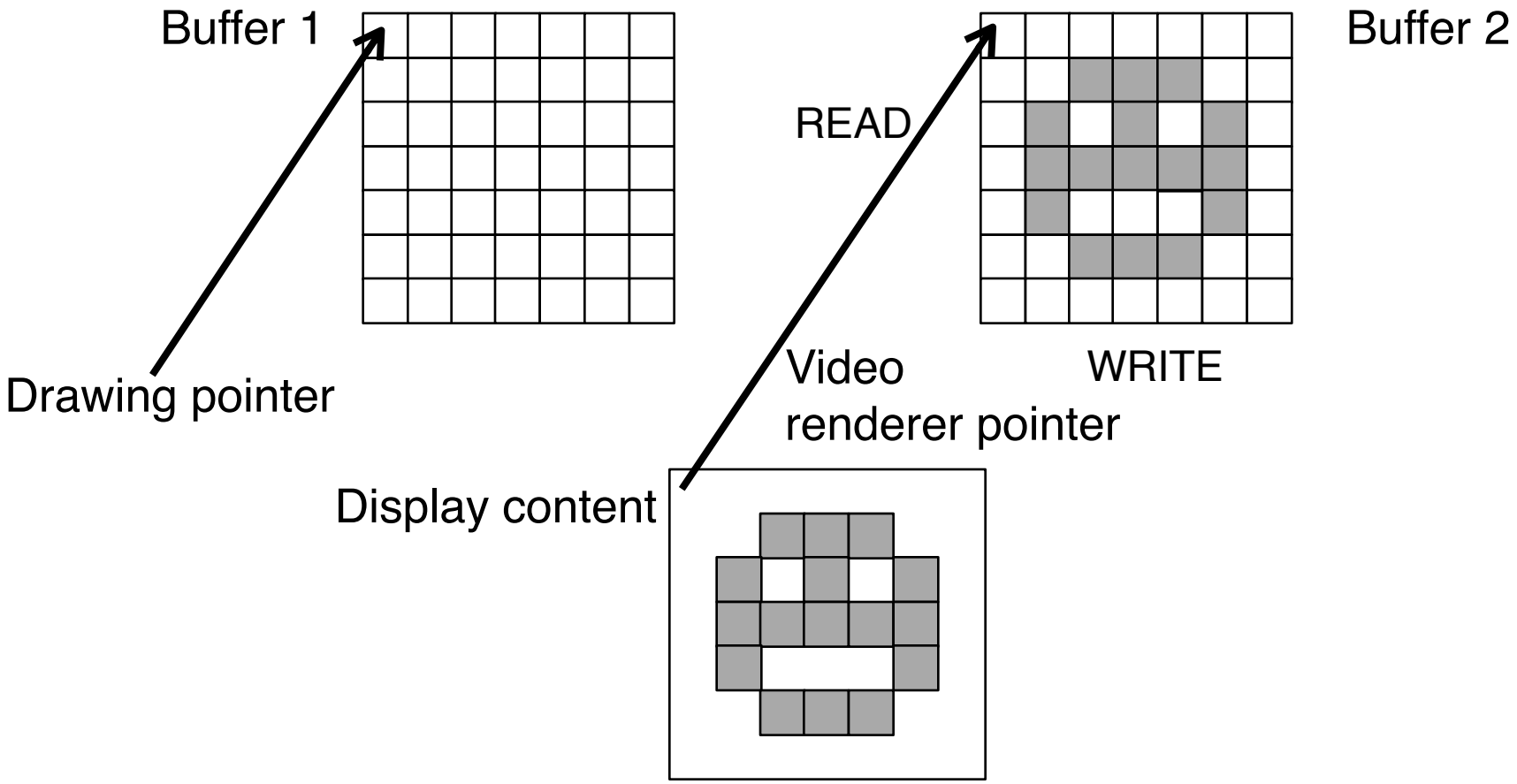
- Using one single graphics buffer
  - Video renderer reads pixel information sequentially and displays it
  - Drawing commands modify same buffer
- Incomplete drawings are rendered to the screen!



Double Buffering: Back Buffer Drawing



Double Buffering: Buffer Flipping



Implementation of Double Buffering

- Basic properties of double buffering
  - Improves user experience
  - Slight cost in speed
- Hardware realization
  - In graphics cards
- Software access
  - Clearly visible in low-level frameworks
  - Mostly hidden in high-level frameworks

Blitting

- BLIT = Block Image Transfer
  - Bit blit = Name for a hardware technology for speeding up image transfer into frame buffer
- Speedup fro drawing:
  - Combine small local changes into a larger buffer
  - Display large image at once – faster than individual updates
- Application for copying whole window contents (moving a window)
- Realized in graphics cards and working together with Double Buffering

Code Examples

- Pygame:

```
slide = pygame.image.load('pics/tiger.jpg').convert()
screen.blit(slide, (50, 50))
pygame.display.update()
```

- SFML:

```
window.clear(bg);
sprite.setTexture(loadFromFile("pics/tiger.jpg"));
window.draw(sprite);
window.display();
```

- Cocos2d-x:
  - Create nodes in scene graph
  - Put scene onto primary stage

Screen/Surface Locking

- *Locking* reserves a part of the screen (*surface* in Pygame)
  - No other process can interfere (“one change at a time, please”)
- Low-level technique:
  Manual locking/unlocking may improve performance
  - Lock/unlock pair of commands around logically contingent group of graphics commands
- Locking is applied automatically in most graphics frameworks
- Example (Pygame):

```
screen.lock()
pygame.draw.rect(screen, red, Rect((10, 10), (230, 80)))
pygame.draw.circle(screen, white, (50, 50), 40)
pygame.draw.circle(screen, white, (200, 50), 40)
screen.unlock()
```

6 Images, Vector Graphics, and Scenes

6.1 Image Buffers

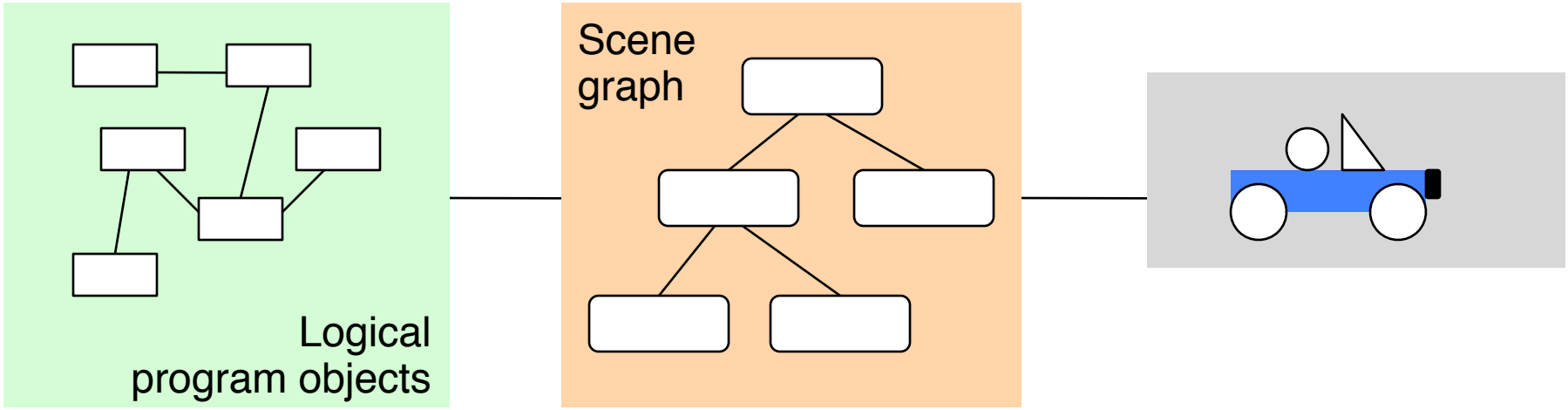
6.2 Structured Graphics: Scene Graphs 

6.3 Sprites

Literature:

B. B. Bederson, J, Grosjean, J. Meyer: Toolkit Design for Interactive Structured Graphics, *IEEE TSE* vol. 30 no. 8, pp. 535-546, 2004

Scenes, Objects and Groups



- *Scene:* Collection of all relevant (view-oriented) objects
  - Abstract representation of the “world” (in a certain state)
- Often several objects are grouped into one (view-oriented) representation
  - Operations shall be applied to whole group (movement, copy, …)
- Two-level view mechanism:
  - Model
  - Scene graph (abstract view)
  - Concrete view

Scene Graphs Are High-Level Constructs

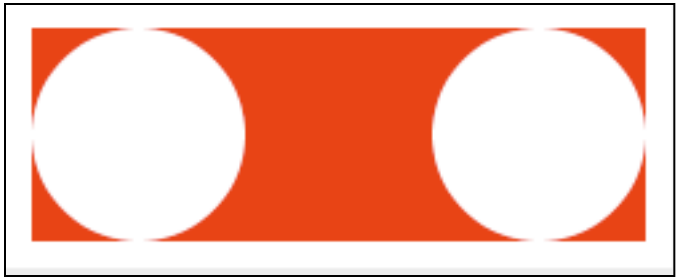
- Pygame, SFML:
  - No built-in scene graph
  - External library or own implementation
  - For Pygame, see for instance:
    http://pygame.org/project-pygext-103-136.html
  - For SFML, see for instance:
    J. Haller, H.V. Hansson, A. Moreira: SFML Game Development, Packt 2013, Chapter 3
- Cocos2d-x, JavaFX:
  - Built-in scene graph
  - Scene graph as basis for displaying anything

Example: Scene Graph for SFML

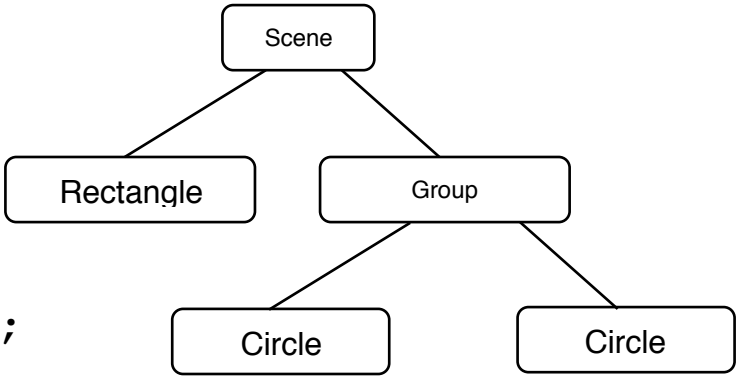
- Excerpt of the implementation in Haller et al. 2013
  - Note: Heavy use of modern C++11

```
class SceneNode {

  public:
    typedef std::unique_ptr<SceneNode> Ptr;

  public:
    SceneNode();

  private:
    std::vector<Ptr> mChildren;
    SceneNode* mParent;

};
```

Reminder: Scene Graph with JavaFX



```
Group root = new Group();
Scene scene = new Scene(root, 250, 100);

Rectangle r = new Rectangle(10, 10, 230, 80);
r.setFill(Color.RED);
root.getChildren().add(r);

Group circles = new Group();
circles.setTranslateX(10);
circles.setTranslateY(10);
root.getChildren().add(circles);

Circle circle1 = new Circle(40, 40, 40);
circle1.setFill(Color.WHITE);
circles.getChildren().add(circle1);

Circle circle2 = new Circle(190, 40, 40);
circle2.setFill(Color.WHITE);
circles.getChildren().add(circle2);

primaryStage.setTitle("JavaFX Scene Graph");
primaryStage.setScene(scene);
```



6 Images, Vector Graphics, and Scenes

6.1 Image Buffers

6.2 Structured Graphics: Scene Graphs

6.3 Sprites 

Literature:

Will McGugan: Beginning Game Development with Python and Pygame, Apress 2007

http://cocos2d-x.org/docs/programmers-guide/3/

Sprite

- A *sprite (Kobold, Geist)* is a movable graphics object which is presented on top of the background image.
  - Mouse pointer images are examples of sprites
- Hardware sprite:
  - Outdated technique for hardware-supported fast display of moving image
- Software sprite:
  - Any moving picture displayed over background
- Game sprites are mostly based on bitmap graphics
  - Reasons: Ease of creation, need for bitmapped end product
- Pygame sprite:
  - Special class designed to display movable game objects
- Cocos2d-x sprite:
  - Special class, objects created from bitmap image
  - Many manipulation functions, like setting anchor point, rotation, scale, skew, color, opacity, sprite outline (rectangle vs. polygon)

Example: Simple Sprite in Pygame

```
class MagSprite(pygame.sprite.Sprite):

    def __init__(self):
        pygame.sprite.Sprite.__init__(self)
        self.image = pygame.image.load(sprite_imgfile)
        self.rect = self.image.get_rect()

    def update(self):
        self.rect.center = pygame.mouse.get_pos()

sprite = MagSprite()
allsprites = pygame.sprite.Group()
allsprites.add(sprite)

while True:
    for event in pygame.event.get():
        ...
        screen.blit(background,(0,0))
        allsprites.update()
        allsprites.draw(screen)
        pygame.display.update()
```

Example: Magnifying Glass in Pygame

```
class MagSprite(pygame.sprite.Sprite):

    def __init__(self):
        pygame.sprite.Sprite.__init__(self)
        self.image = pygame.image.load(sprite_imgfile)
        self.rect = self.image.get_rect()

    def update(self):
        (mx,my) = pygame.mouse.get_pos()
        self.rect.center = (mx,my)
        if (mx+d/2 < 256) and (my+d/2 < 256)
         and (mx-d/2 > 0) and (my-d/2 > 0):
            magview = background.subsurface(
              (mx-d/2,my-d/2,d,d)).copy()
            magview = pygame.transform.scale(magview,(256,256))
            screen.blit(magview,(256,0))
```

Example: Simple Sprite in Cocos2d-x (1)

- Creating a sprite:

```
cocos2d::Sprite* cursorSprite = Sprite::create();
this->addChild(cursorSprite);
```

- Listening to mouse movements and placing the sprite:

```
auto listener = cocos2d::EventListenerMouse::create();
listener->onMouseMove = [=](Event* event) {
  EventMouse* e = (EventMouse*)event;
  cursorSprite->setPosition
     (Vec2(e->getCursorX(), e->getCursorY()));
 };
```

- Please note:
  - Event listener needs to be registered separately (see next slide)!
  - Lambda declaration is just one way to write the code

Example: Simple Sprite in Cocos2d-x (2)

- Listening to mouse clicks and executing an action:

```
listener->onMouseUp = [=](Event* event) {
  EventMouse* e = (EventMouse*)event;
  //… for instance, create a new sprite …
  cocos2d::Sprite* imageSprite = Sprite::create();
  this->addChild(imageSprite);
  …
};
```

- Registering the listeners(!):

```
imageFrame->getEventDispatcher()
 ->addEventListenerWithSceneGraphPriority(listener, imageFrame);
```