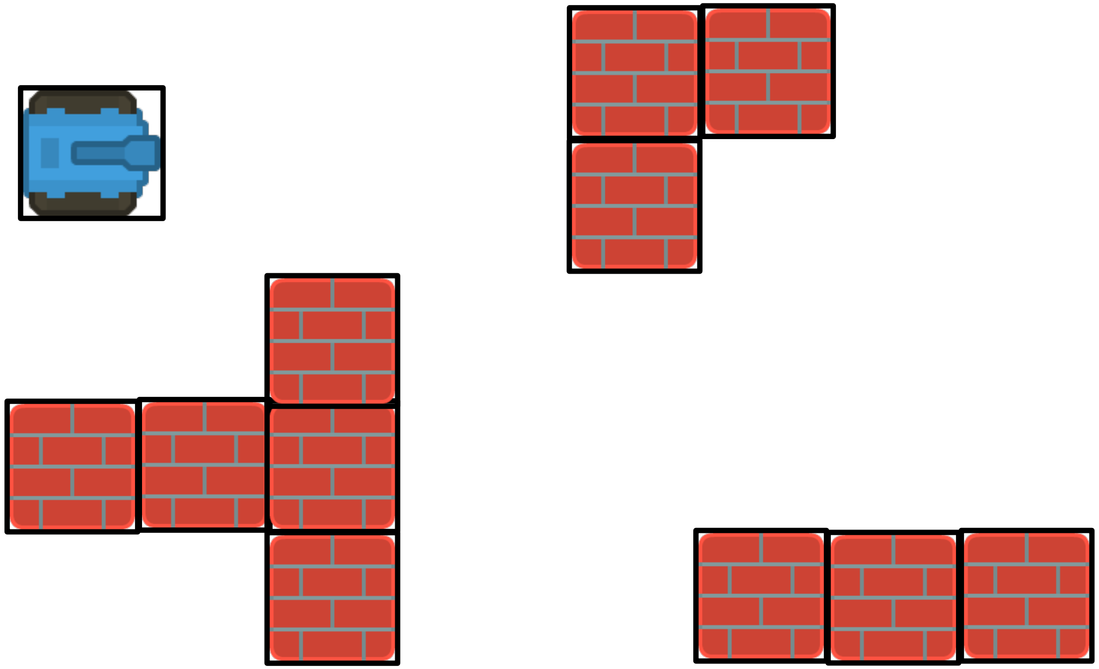
Introduction to Processing

Top-Down Games

Top-Down Games

Top-down games, also sometimes referred to as **bird's-eye view games**, refers to games where the camera angle that shows players and the areas around them is directly above.



Collision Detection

Assume that we already implemented the two collision detection methods below:

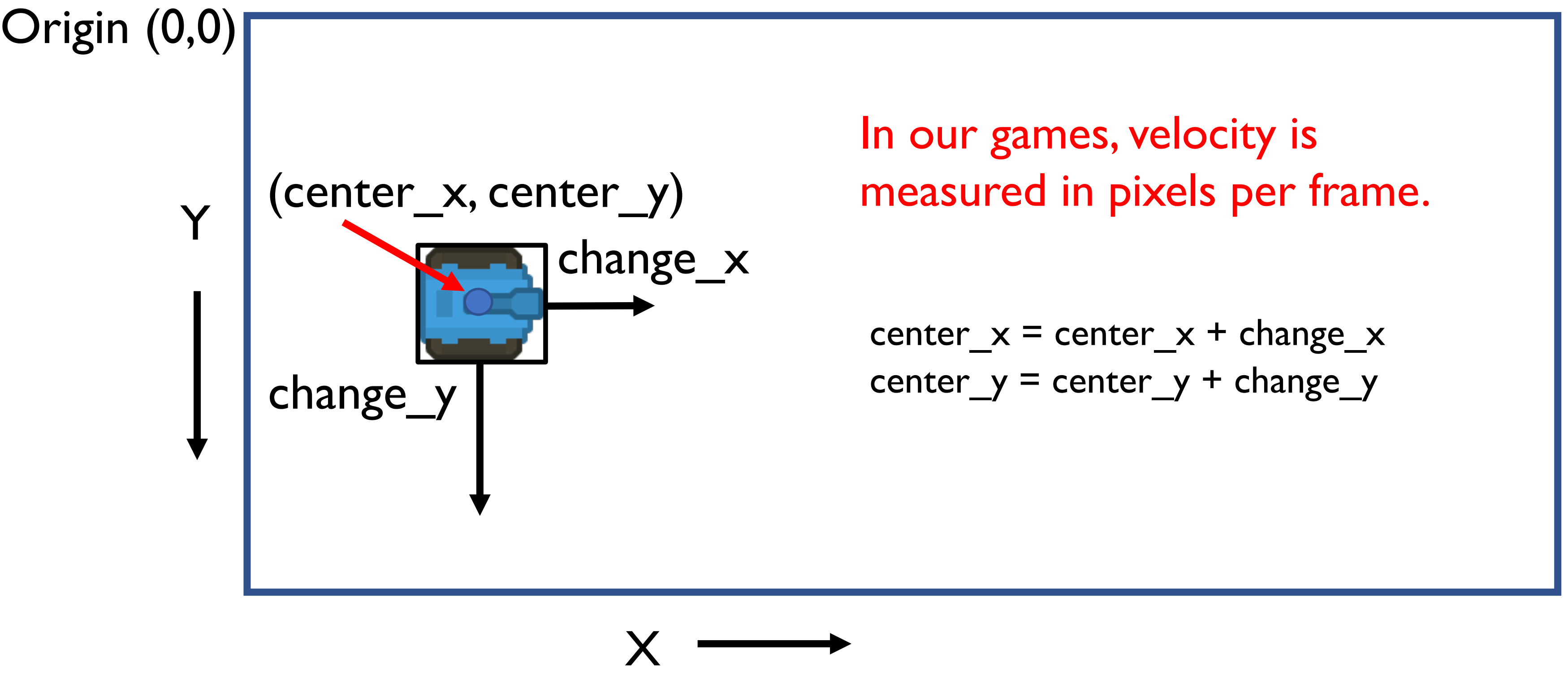
```
def check_for_collision(sprite1, sprite2):
    # returns whether sprite1 and sprite2 intersects


def check_for_collision_list(sprite, sprite_list):
    #returns list of sprites in sprite_list which
    #intersects with sprite.
```

Velocity

Velocity of an object is the rate of change of its position. It is a vector and can be decomposed into a x-component and a y-component.

A Sprite object has attributes change_x and change_y for its velocity.

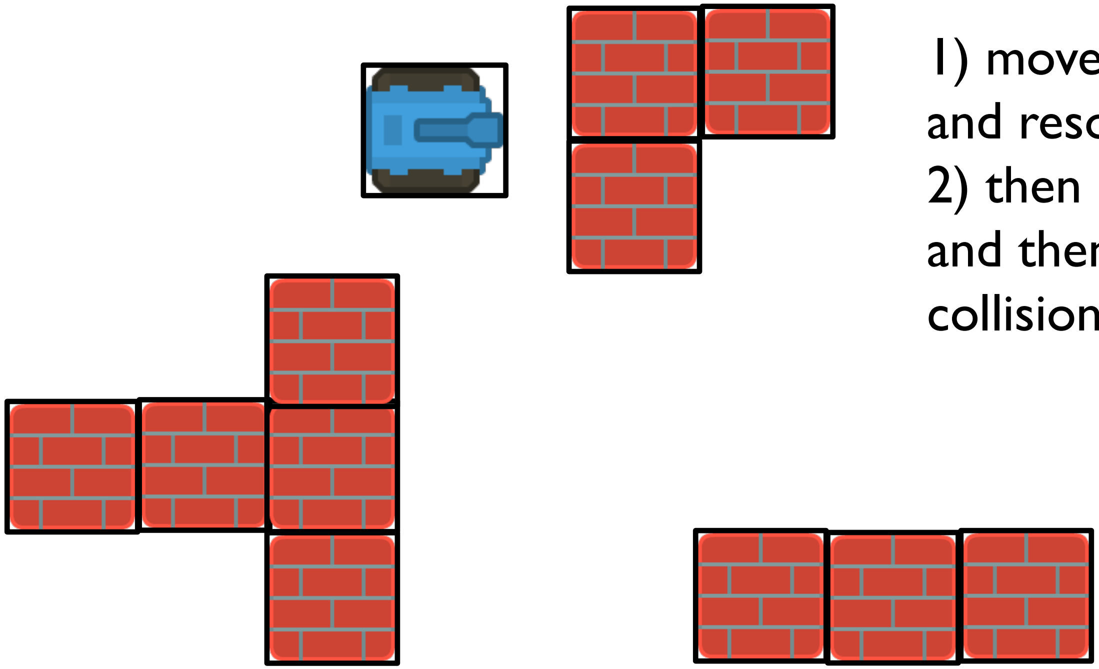


In our games, velocity is measured in pixels per frame.

center_x = center_x + change_x
center_y = center_y + change_y

Resolving Top-Down Collisions

```
center_x += change_x
center_y += change_y
```



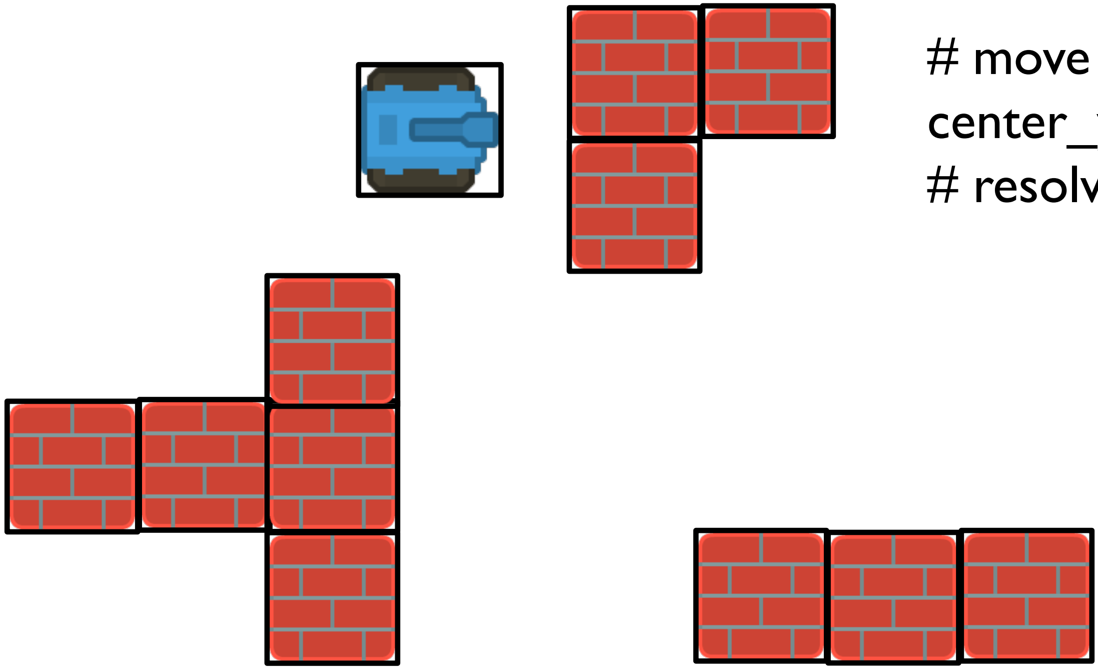
Instead of moving in both the x and y directions and then try to resolve collisions, it is easier to

1) move in x direction, check for and resolve collision
2) then move in the y direction and then check for and resolve collision again.

Resolving Top-Down Collisions

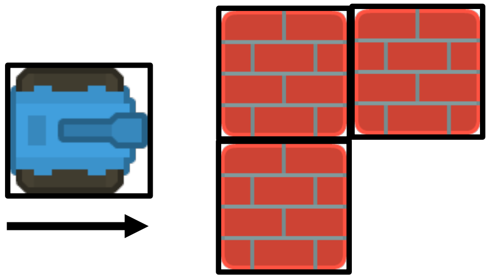
```
# move in horizontal direction
center_x += change_x
# resolve collisions

# move in vertical direction
center_y += change_y
# resolve collisions
```



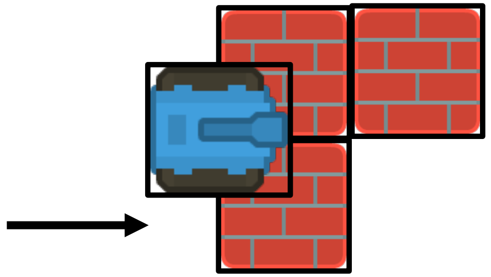
Horizontal Direction

move in horizontal direction

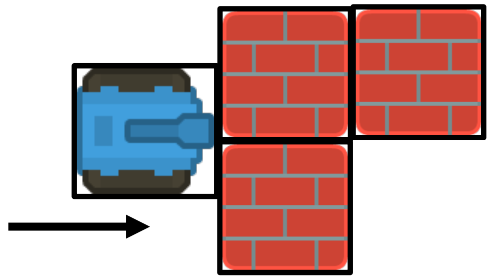


Horizontal Direction

```
move in horizontal direction
compute list of all platforms which collide with playe
if list not empty:
   if player is moving right:
```

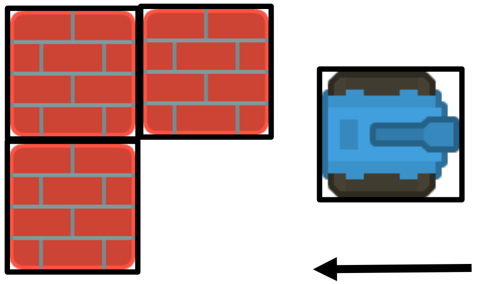


Horizontal Direction



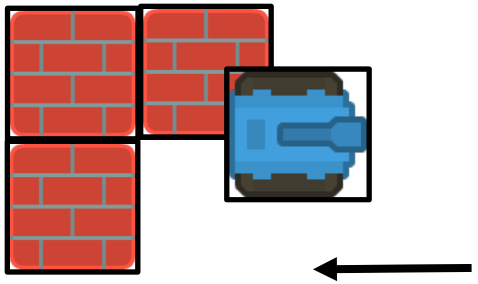
```
move in horizontal direction
compute list of all platforms which collide with playe
if list not empty:
   if player is moving right:
         set right side of player = left side of a
                                    collided platform
```

Horizontal Direction



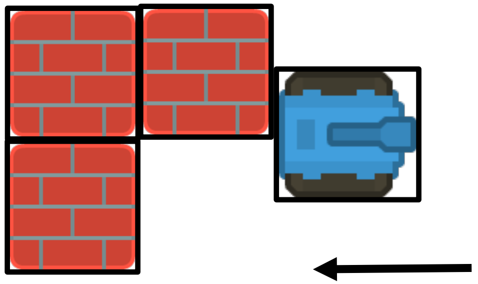
```
move in horizontal direction
compute list of all platforms which collide with playe
if list not empty:
   if player is moving right:
           set right side of player = left side of a
                                        collided platform
  if player is moving left:
```

Horizontal Direction



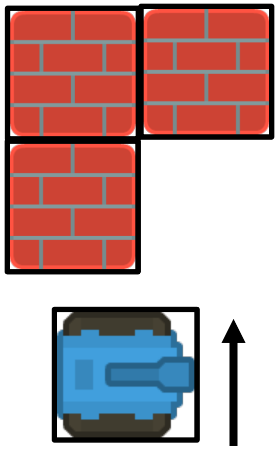
```
move in horizontal direction
compute list of all platforms which collide with playe
if list not empty:
   if player is moving right:
           set right side of player = left side of a
                                          collided platform
  if player is moving left:
```

Horizontal Direction



```
move in horizontal direction
compute list of all platforms which collide with playe
if list not empty:
   if player is moving right:
           set right side of player = left side of a
                                               collided platform
  if player is moving left:
           set left side of player = right side of a
                                               collided platform
```

Vertical Direction



Similarly for the vertical direction:

```
move in vertical direction
compute list of all platforms which collide with player
if list not empty:
    if player is moving up:
        set top side of player = bottom side of a
                                  collided platform
    if player is moving down:
        set bottom side of player = top side of a
                                  collided platform
```