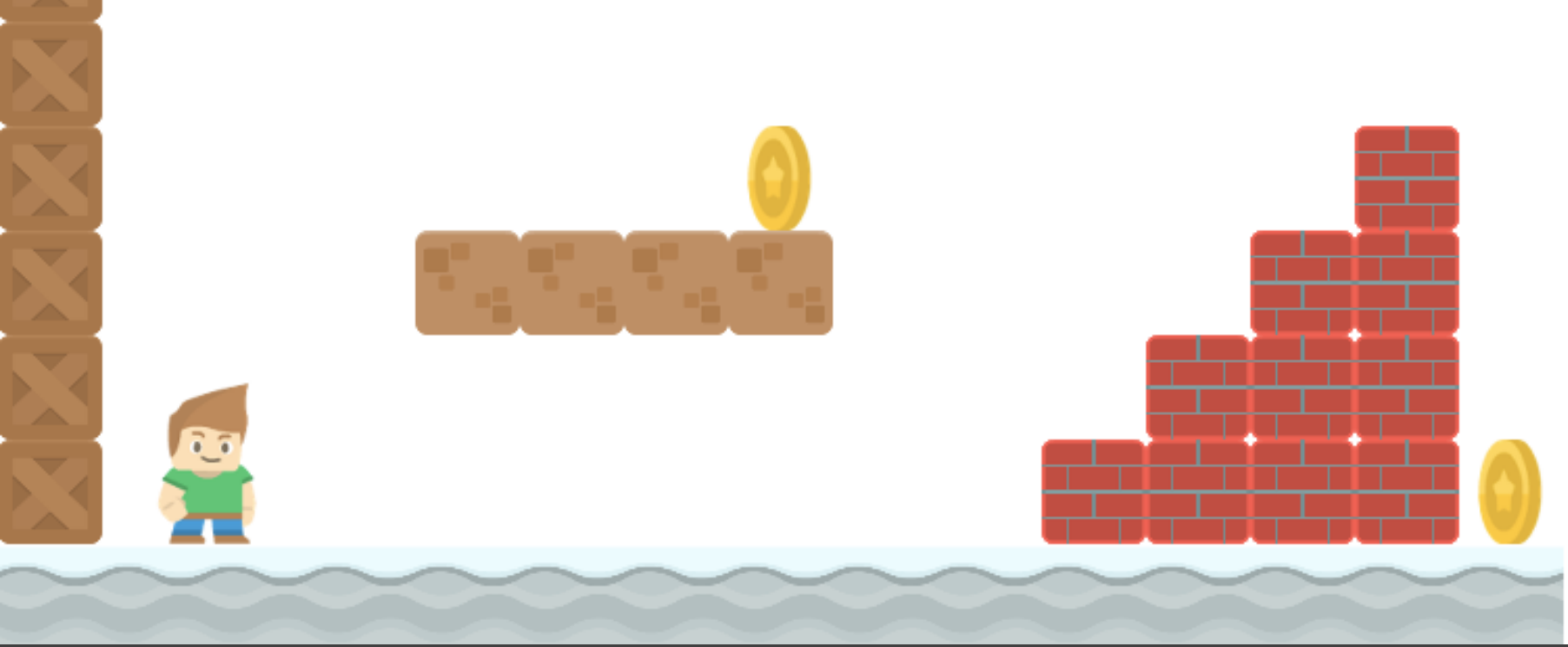
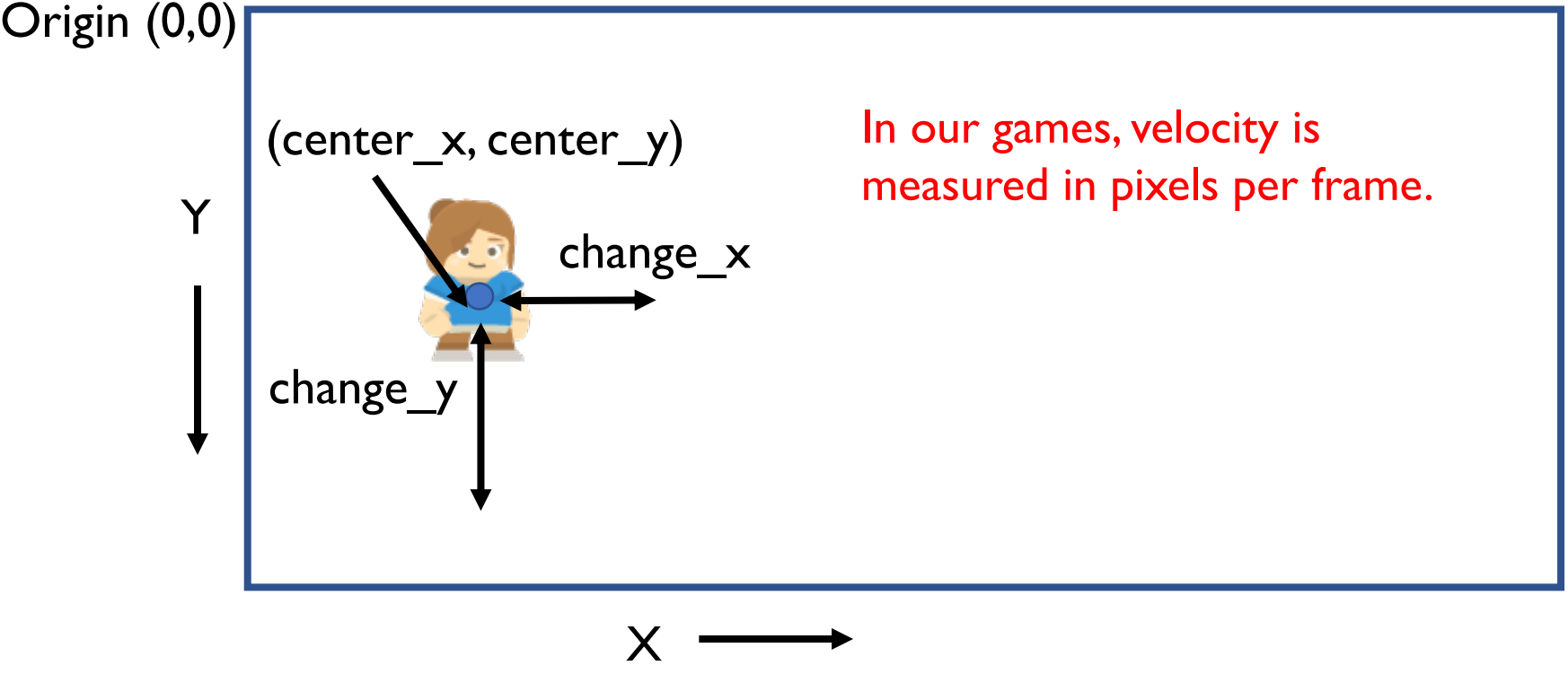


How to Write a Platformer Game in Java

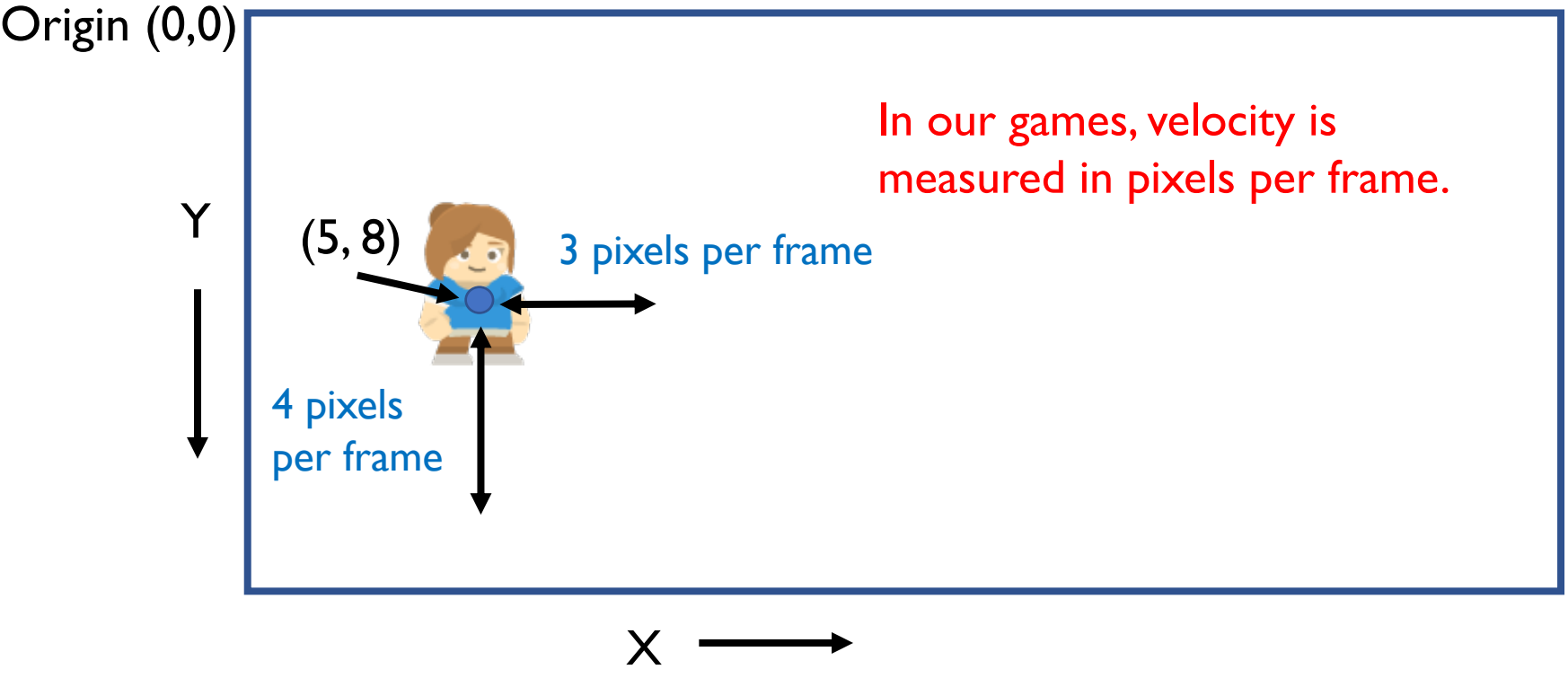
Some Basic Physics

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.

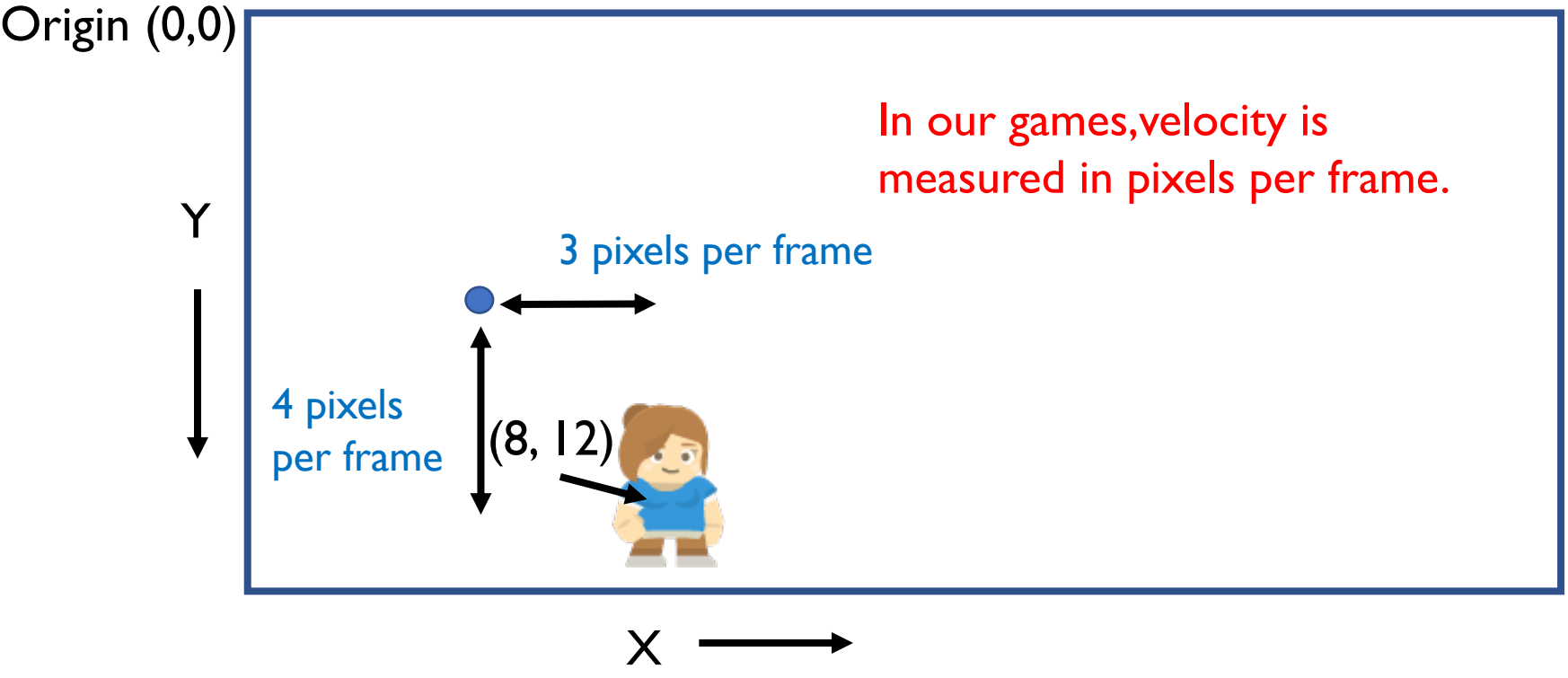
A Sprite object has attributes change_x and change_y for its velocity.



Frame 1

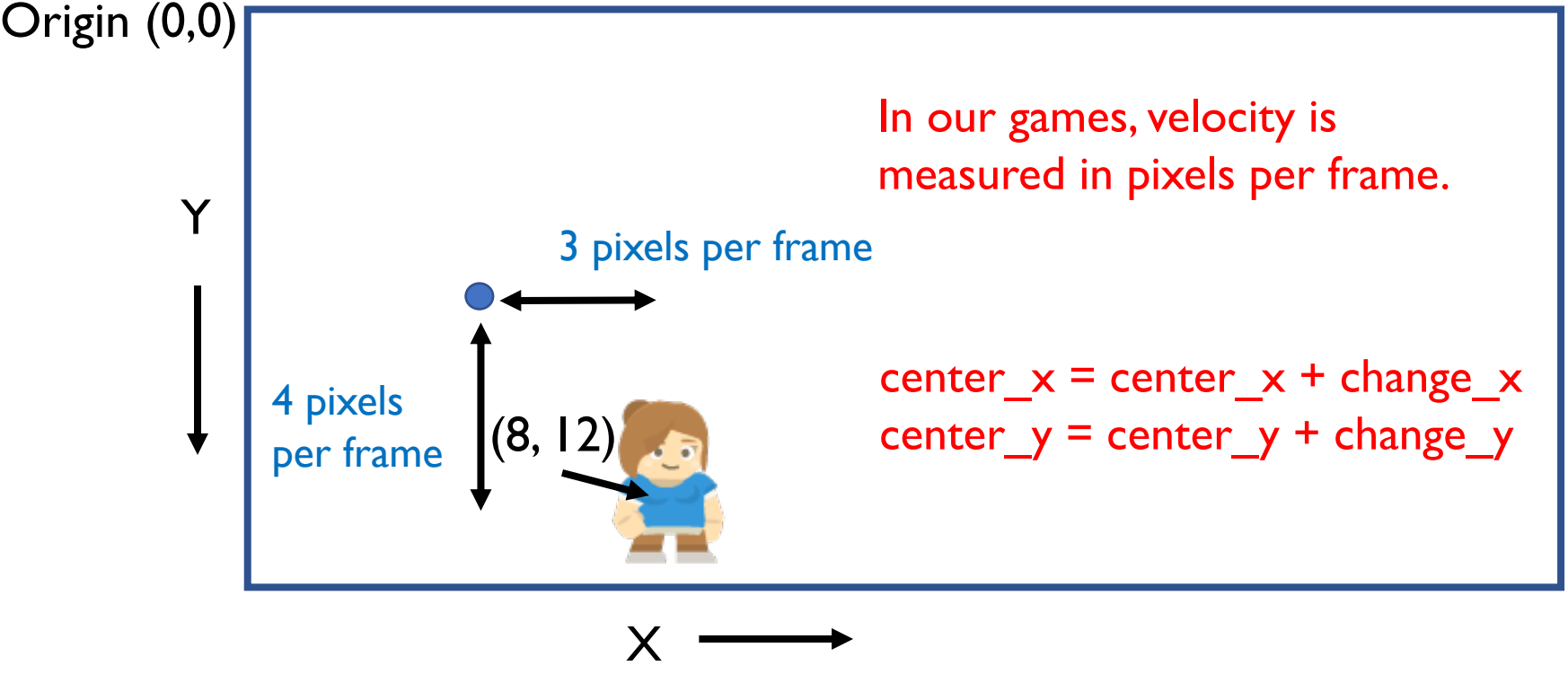

Origin (0,0)
Y
X
(5, 8)
3 pixels per frame
4 pixels
per frame
In our games, velocity is
measured in pixels per frame.

Frame 2


Origin (0,0)
In our games,velocity is measured in pixels per frame.
Y
3 pixels per frame
4 pixels per frame
(8, 12)
X

Position and Velocity

New Position = Old Position + Velocity



Velocity

The velocity of an object is the rate of change of its position.

New Position = Old Position + Velocity

center_x = center_x + change_x
center_y = center_y + change_y

Acceleration

The acceleration of an object is the rate of change of its velocity.

New Velocity = Old Velocity + Acceleration

change_x = change_x + acceleration_x
change_y = change_y + acceleration_y

For us, we will only have acceleration in the y-direction in the form of gravity.

change_y += gravity

Putting it Together

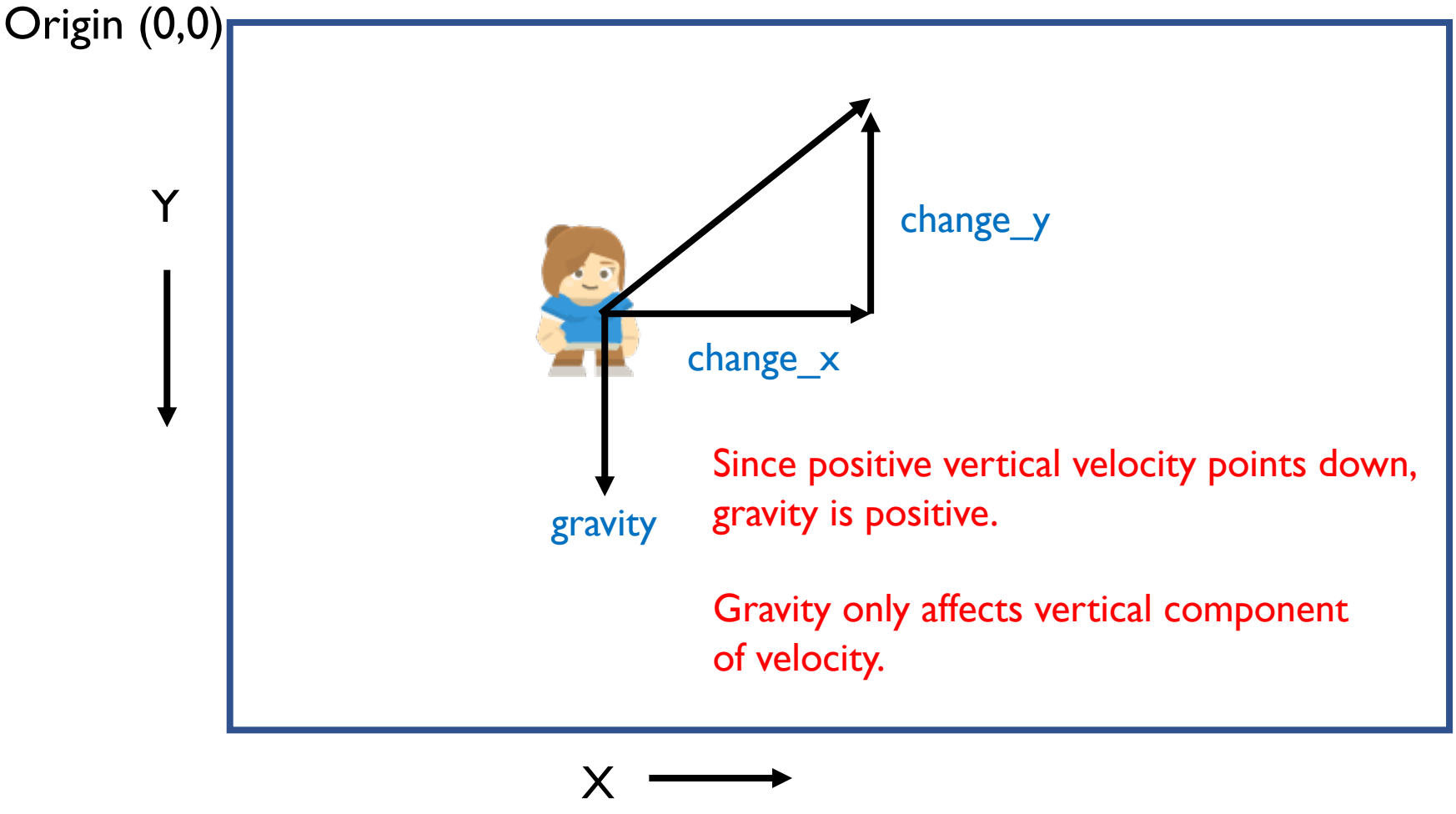
Thus, we just have three very simple formulas:

change_y += gravity

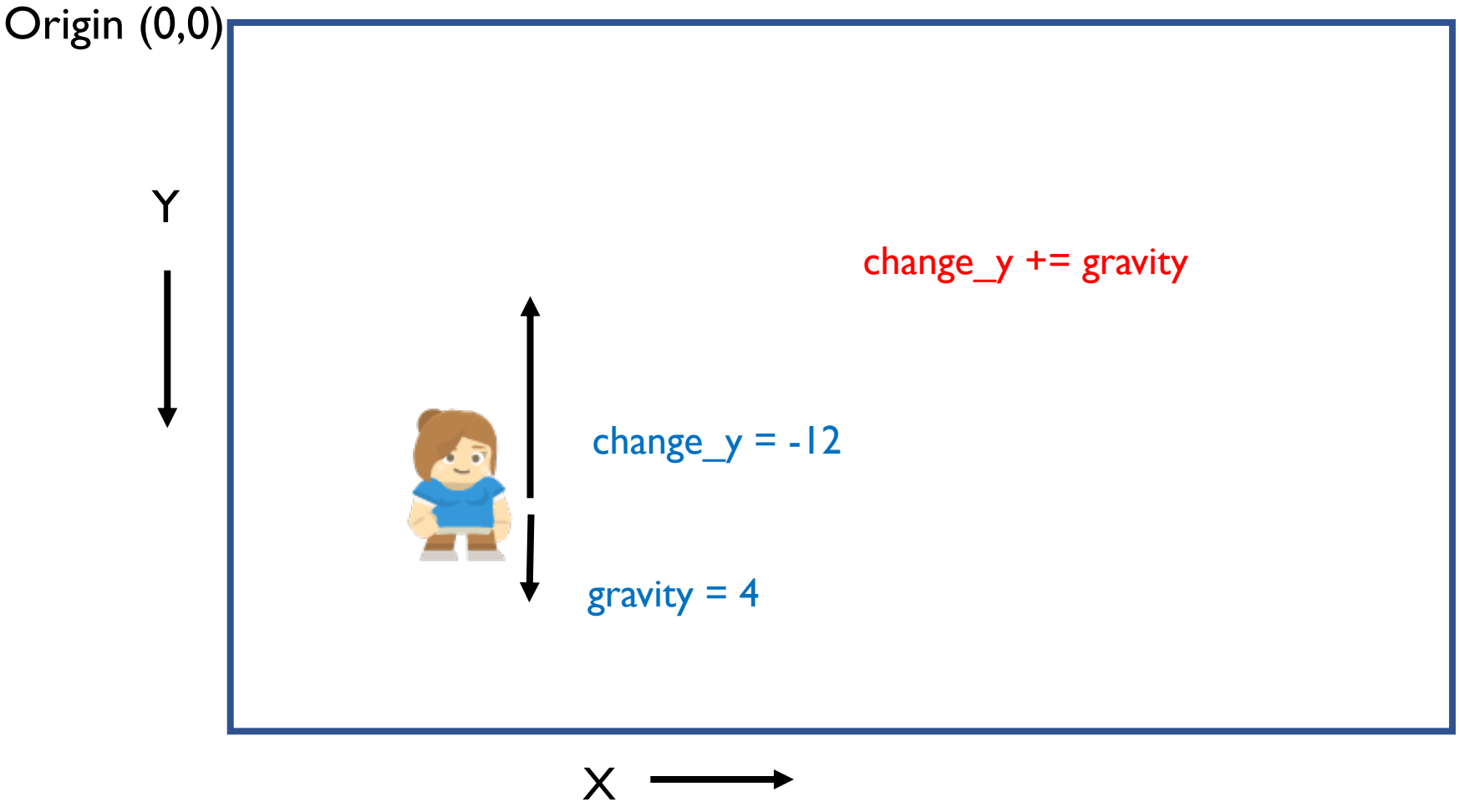
center_y += change_y

center_x += change_x

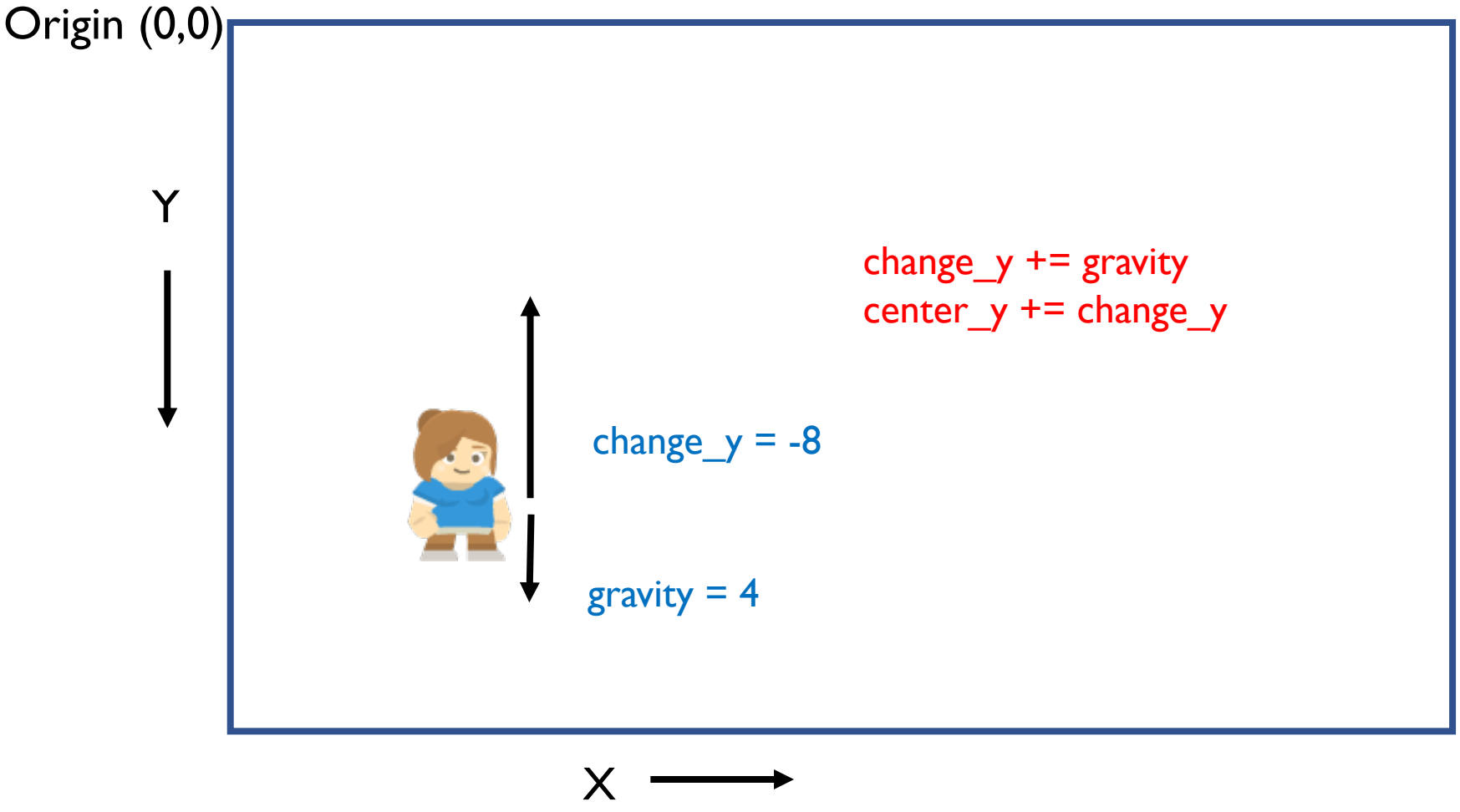
Position and Velocity



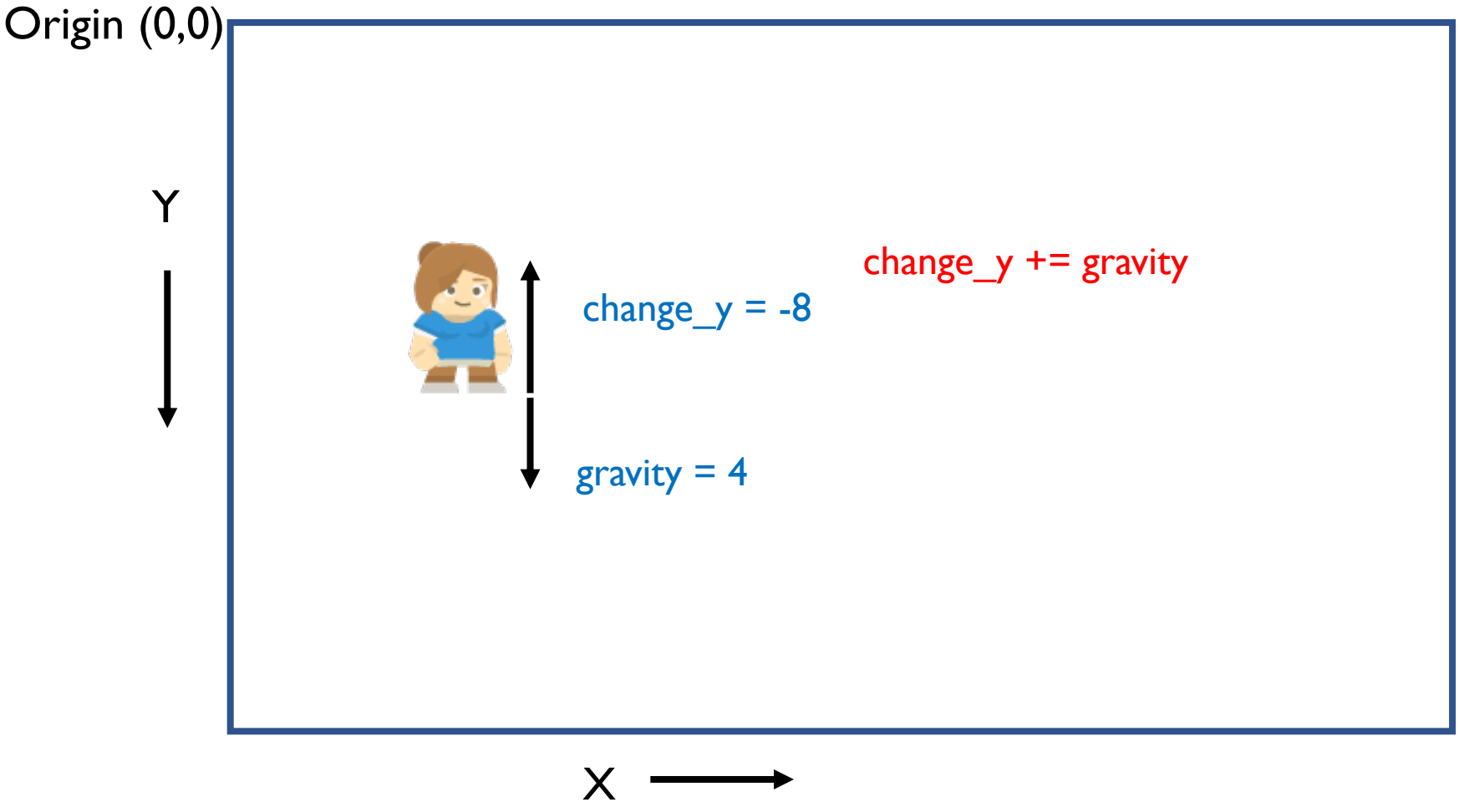
Frame 1



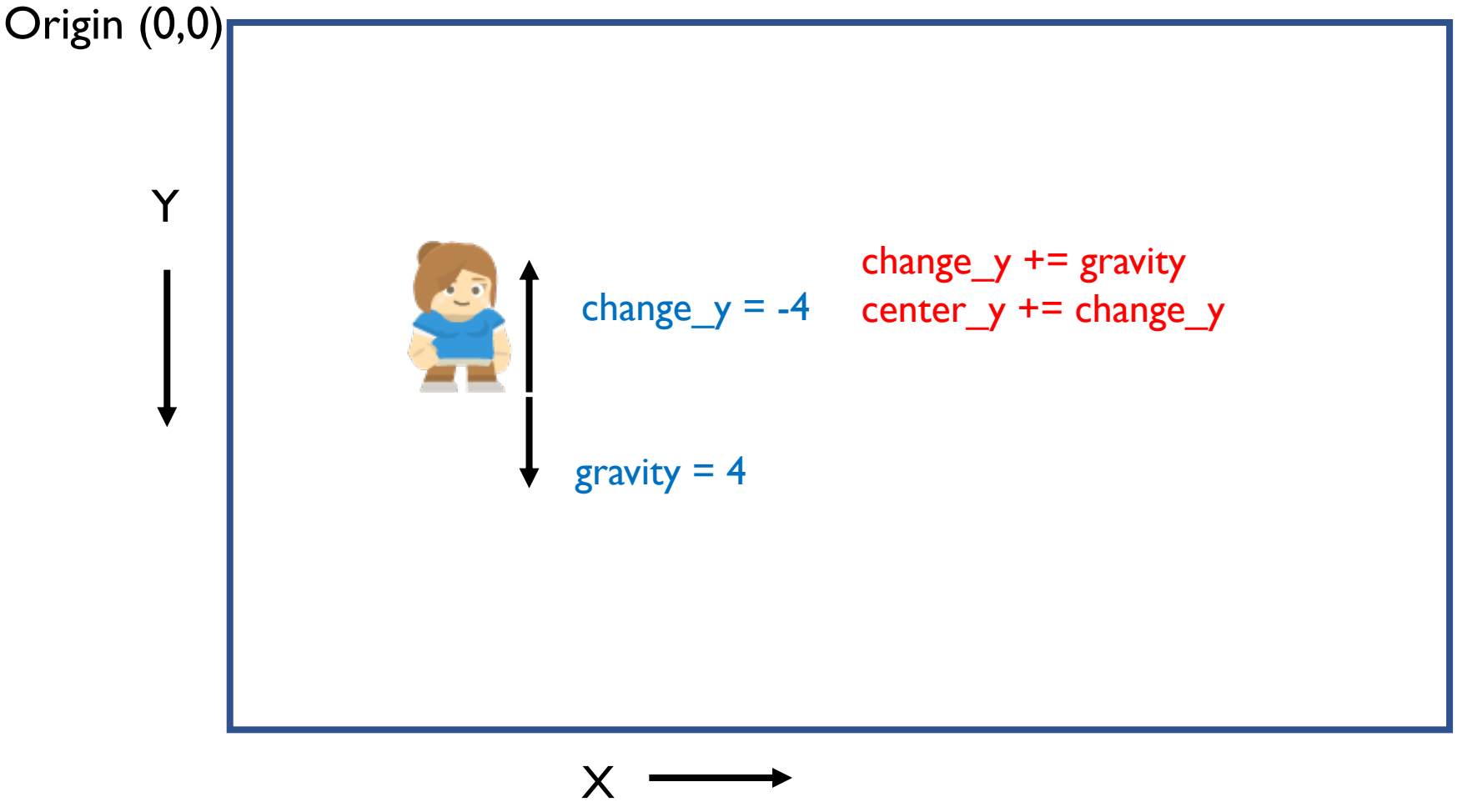
Frame 1


Origin (0,0)
Y
X
change_y += gravity
center_y += change_y
change_y = -8
gravity = 4

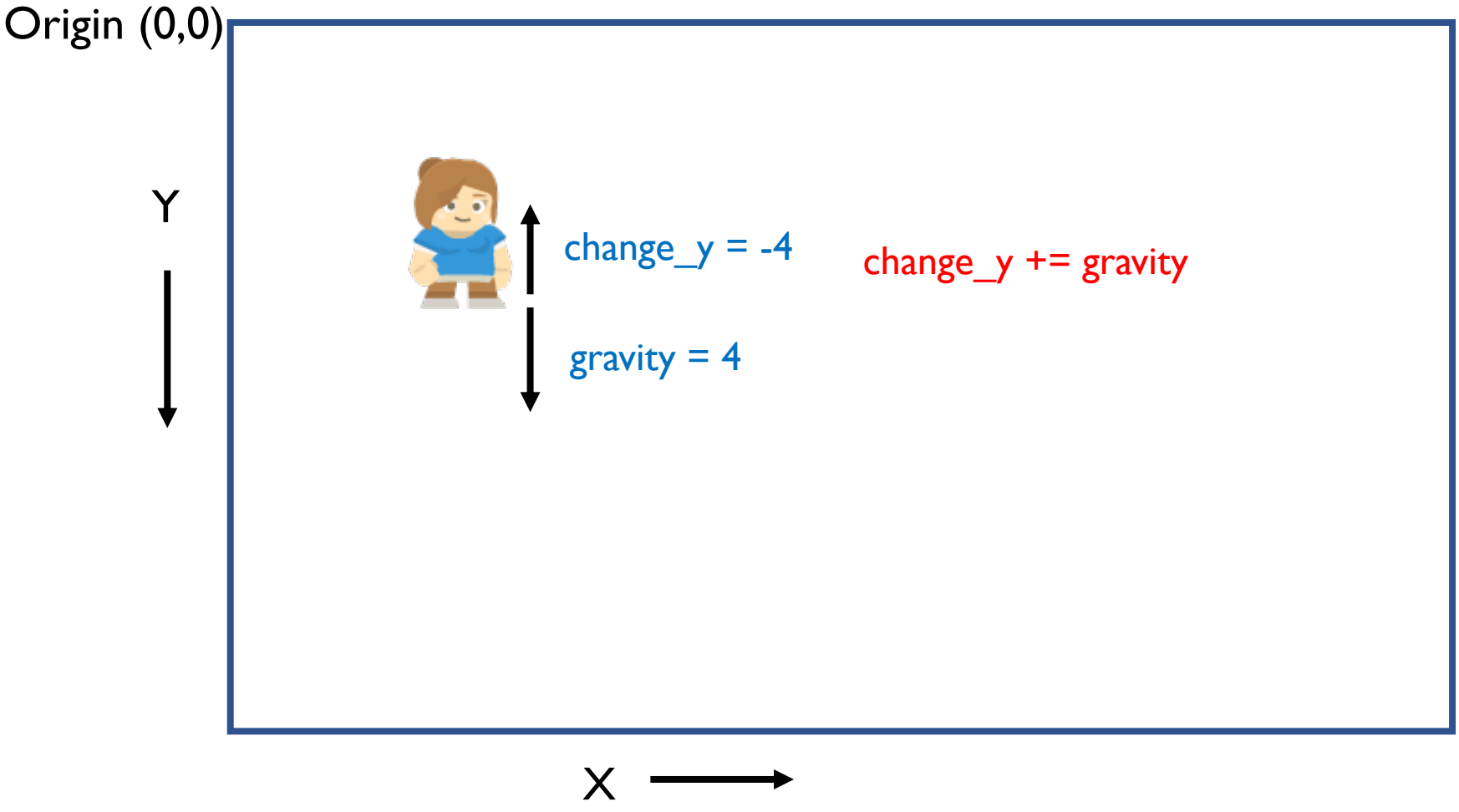
Frame 2



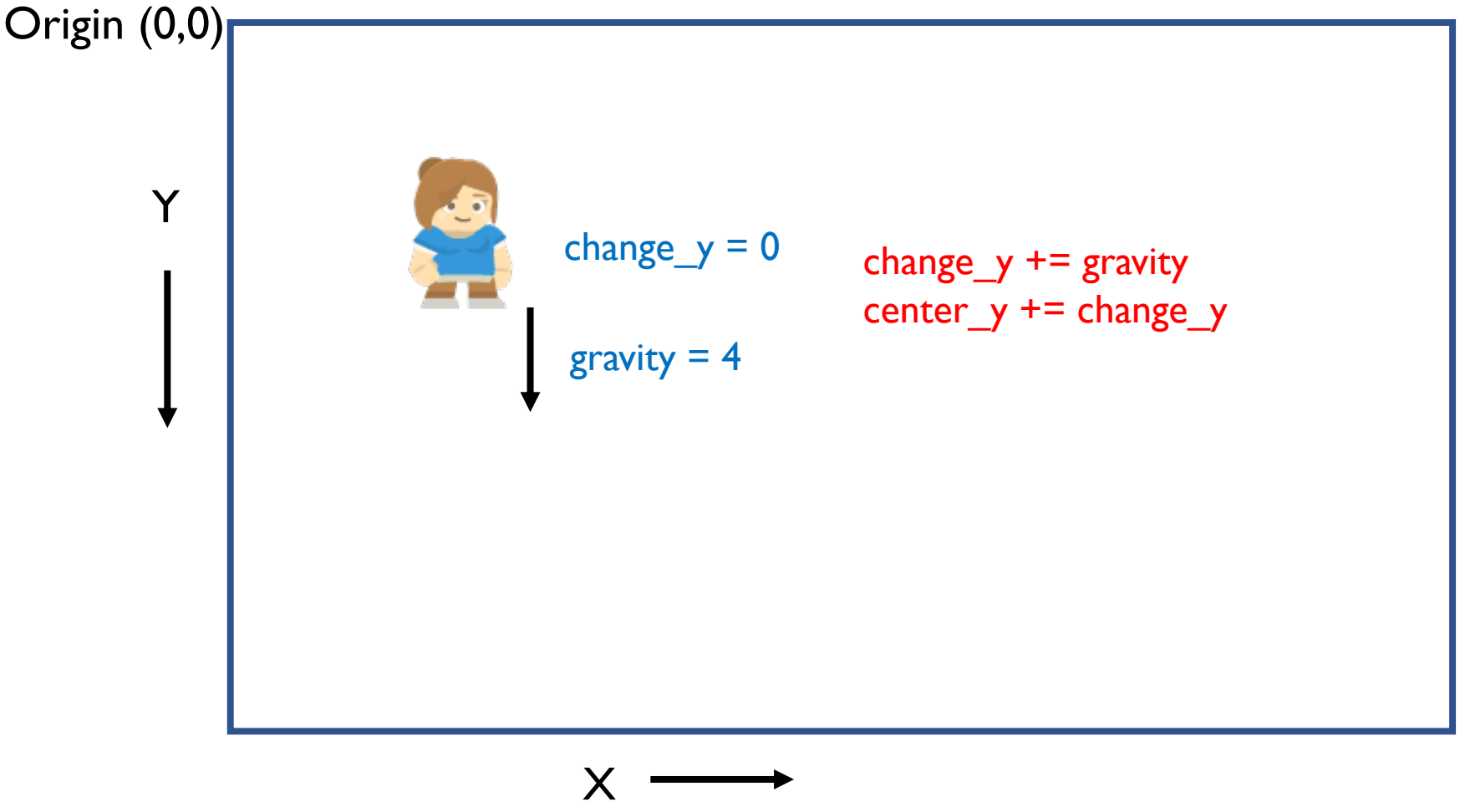
Frame 2


Origin (0,0)
Y
X
change_y = -4
gravity = 4
change_y += gravity
center_y += change_y

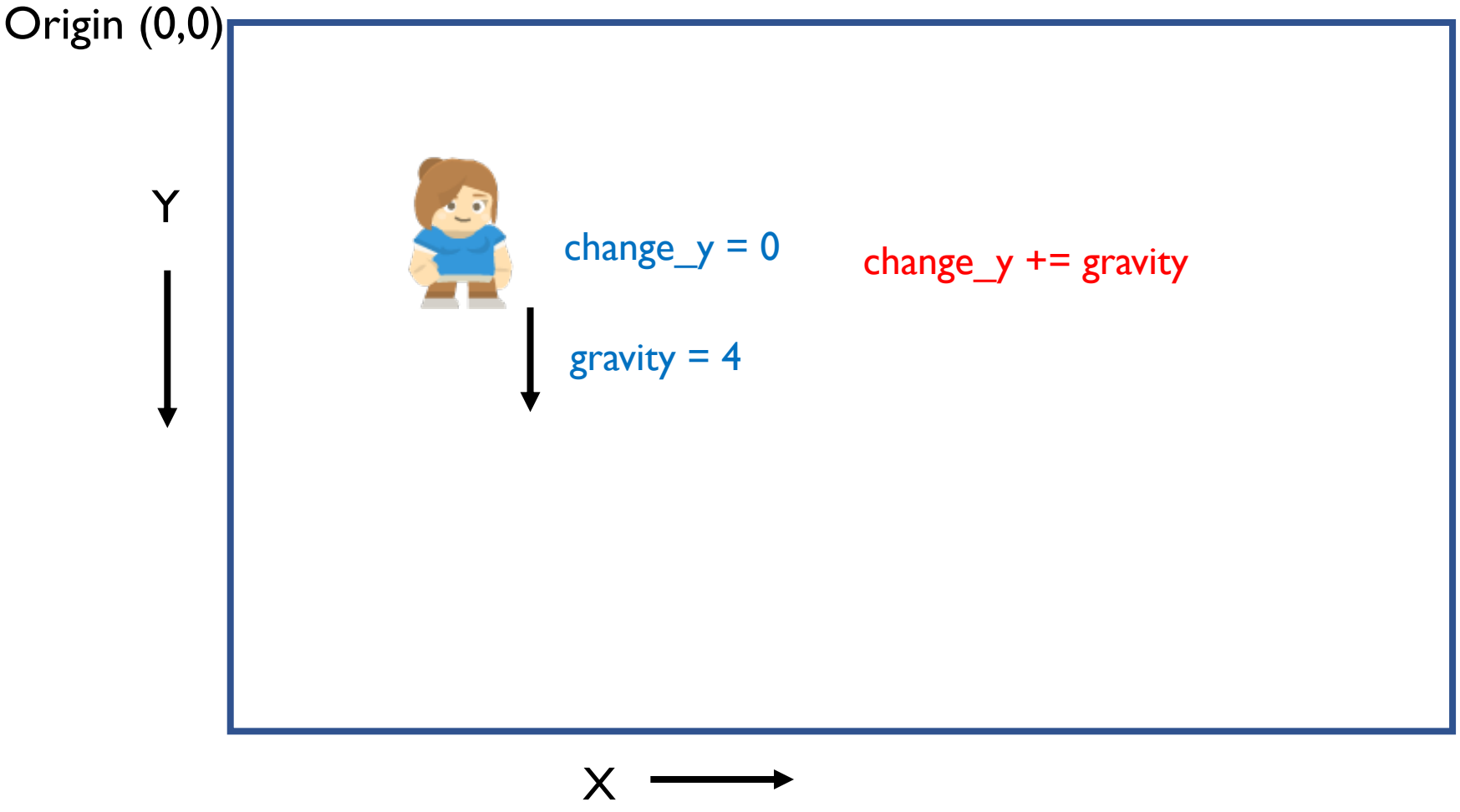
Frame 3



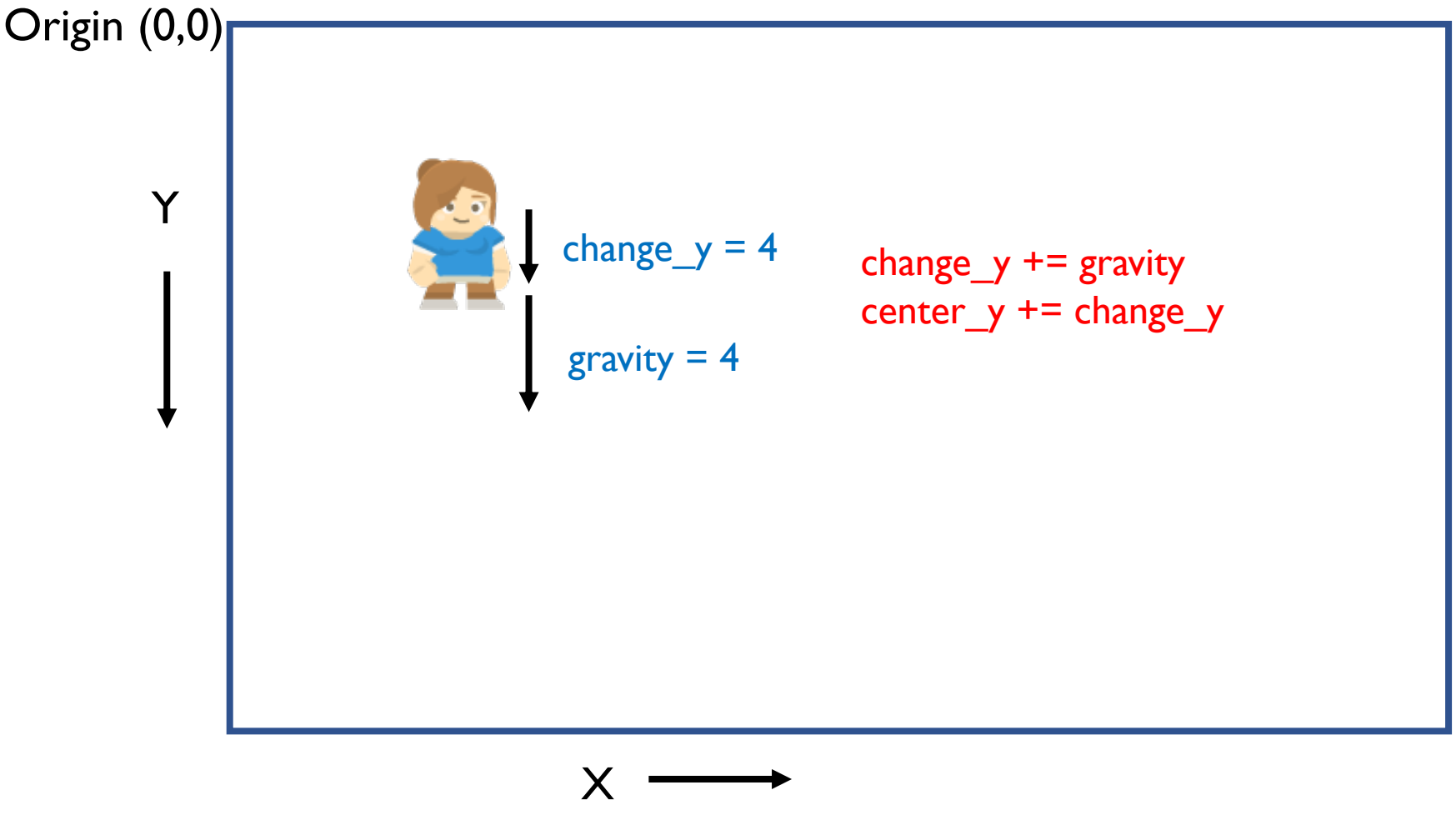
Frame 3


Origin (0,0)
Y
change_y = 0
gravity = 4
change_y += gravity
center_y += change_y
X

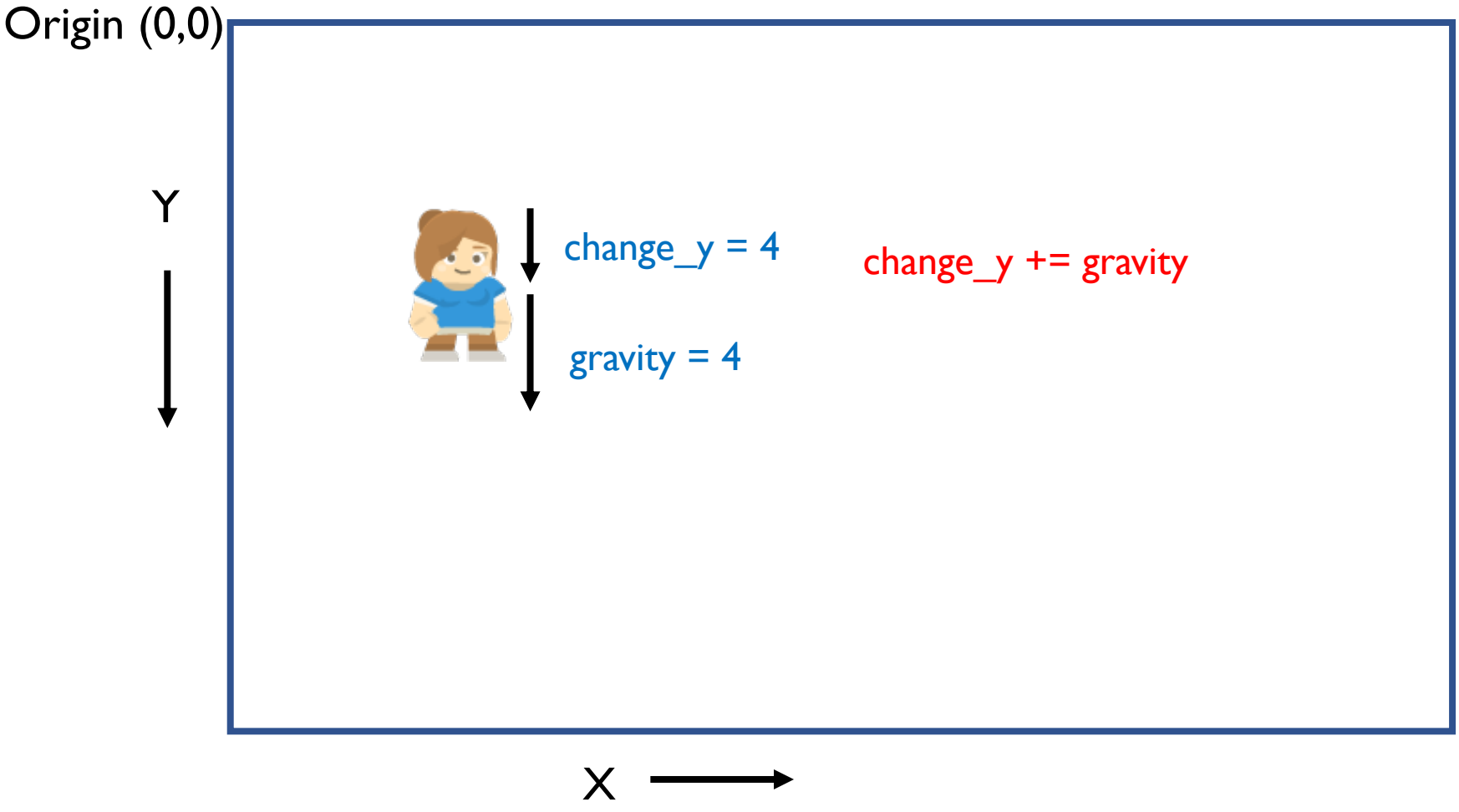
Frame 4



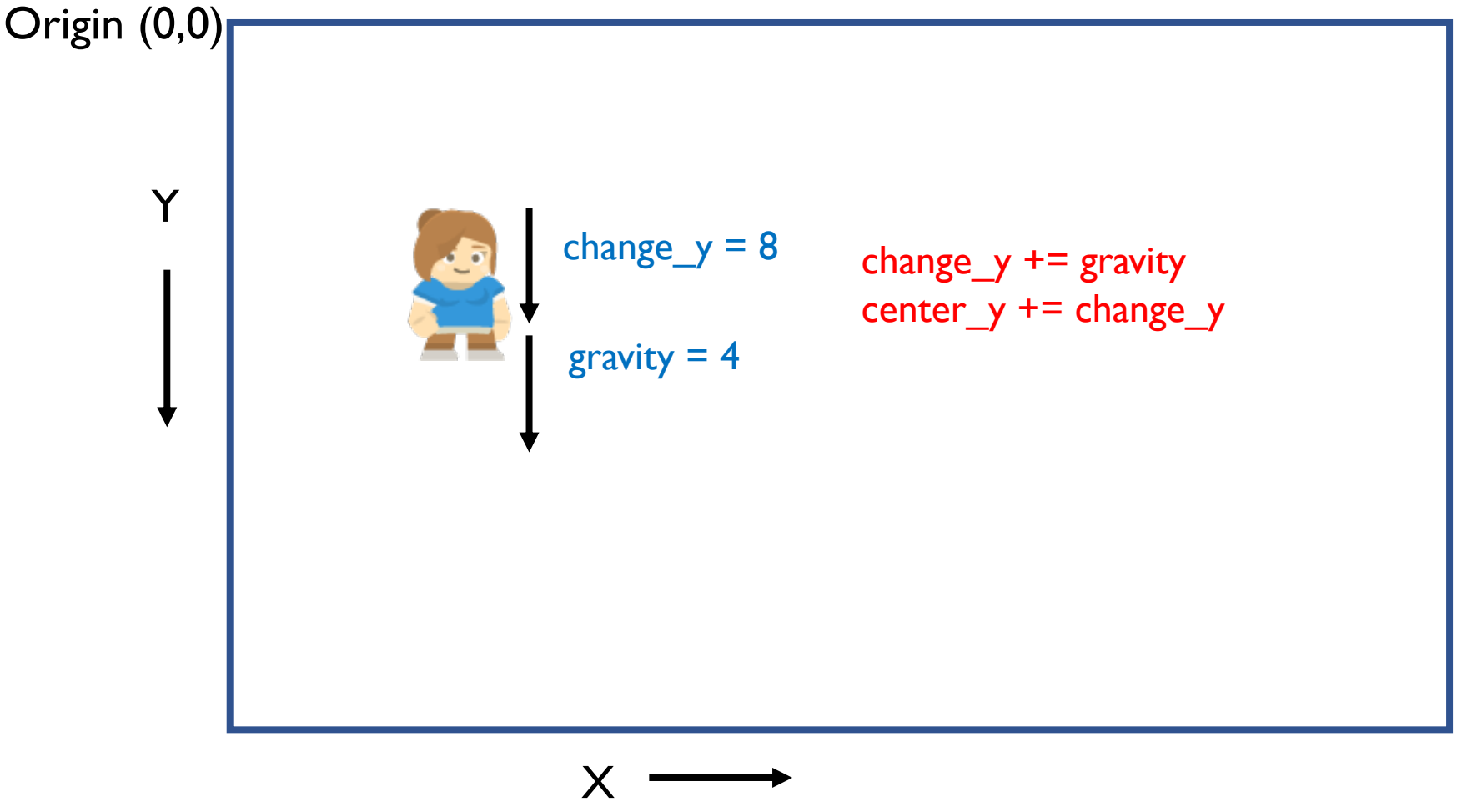
Frame 4


Origin (0,0)
Y
change_y = 4
gravity = 4
change_y += gravity
center_y += change_y
X

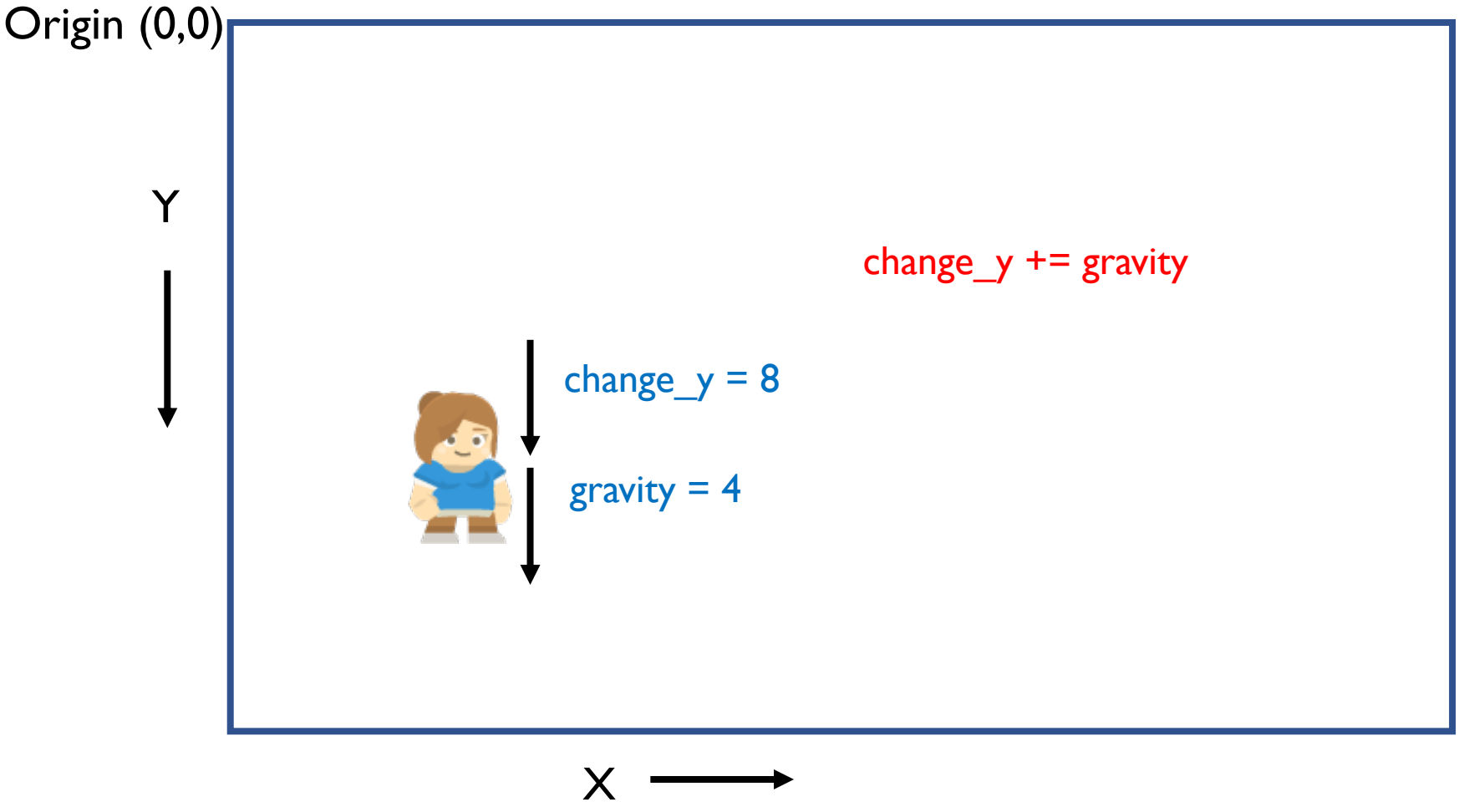
Frame 5



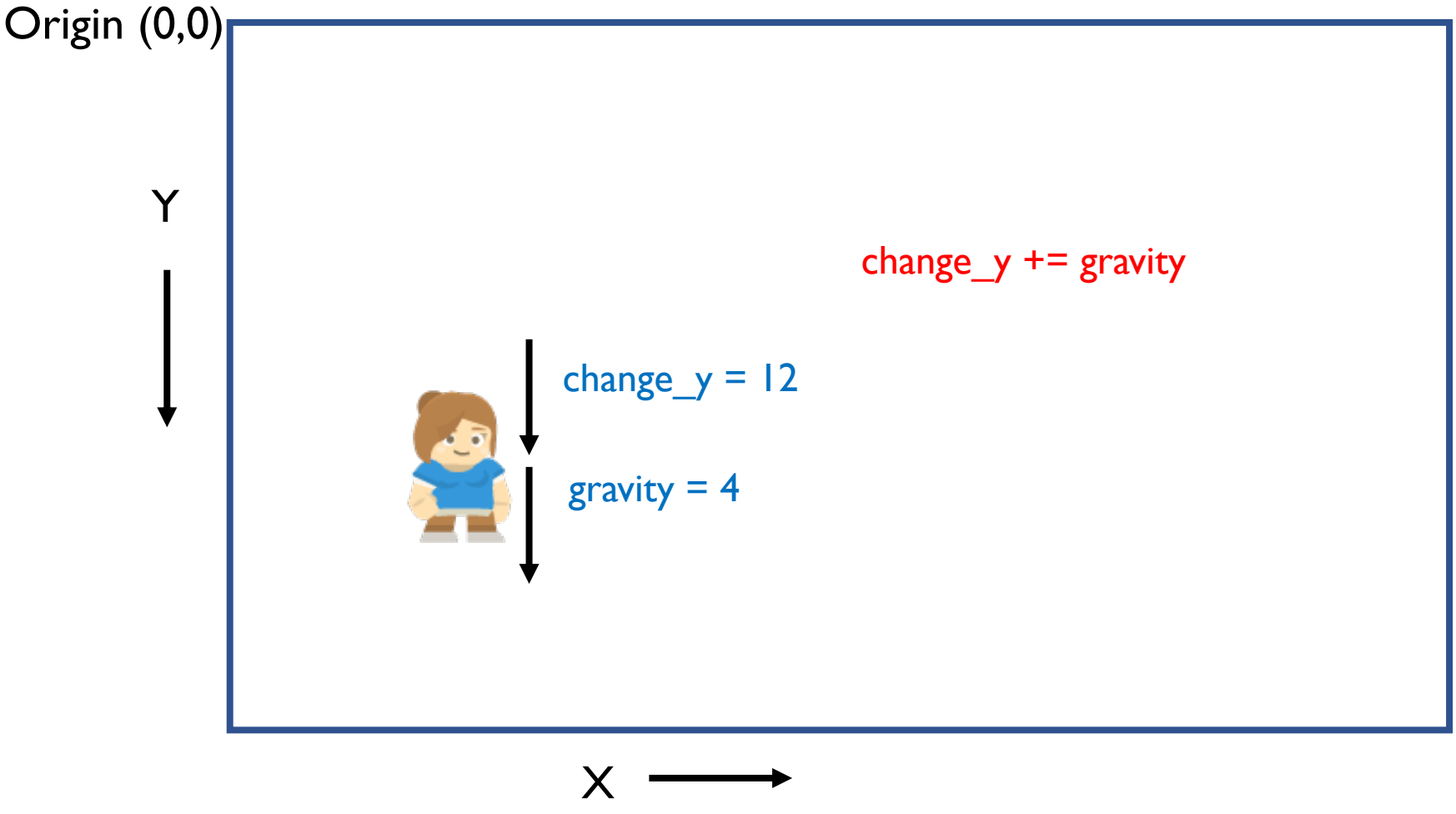
Frame 5


Origin (0,0)
Y
change_y = 8
gravity = 4
change_y += gravity
center_y += change_y
X

Frame 6

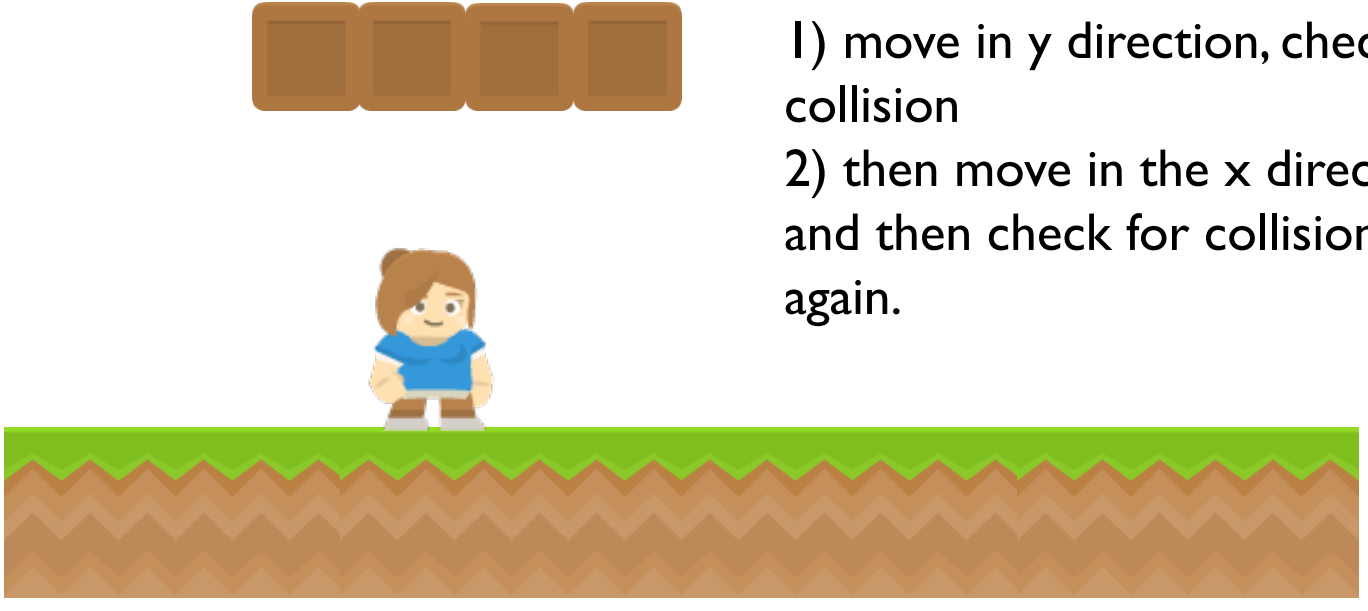


Frame 6



Resolving Platform Collisions

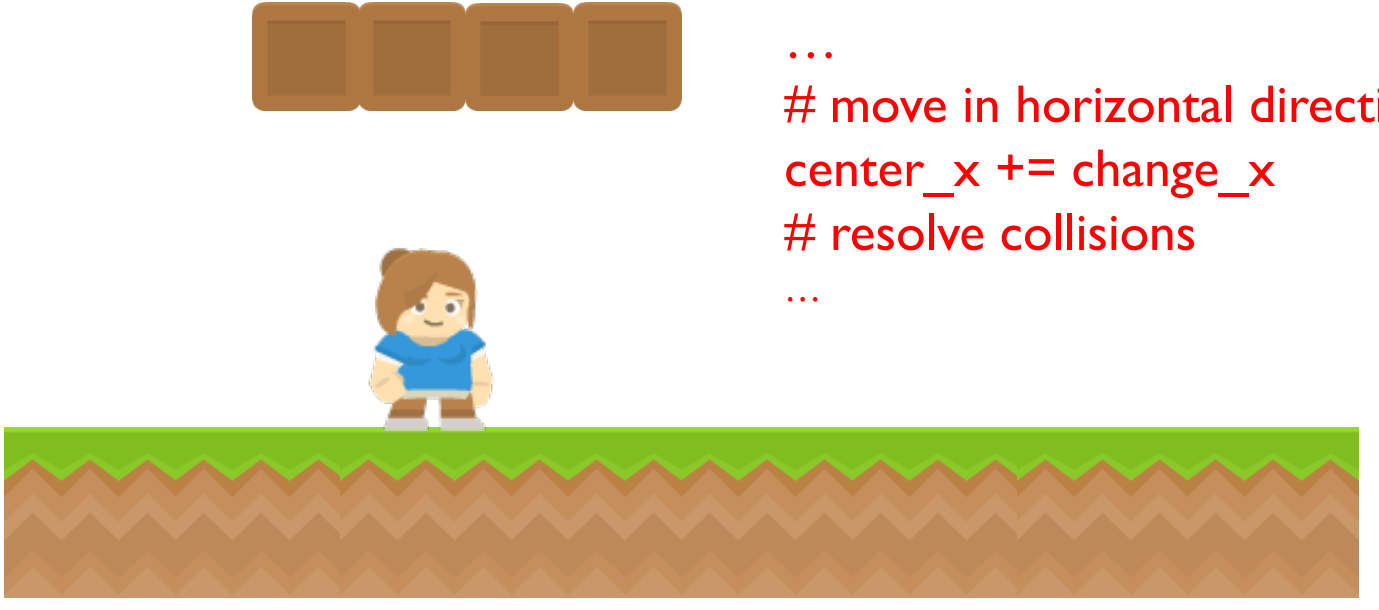
```
change_y += gravity
center_y += change_y
center_x += change_x
```



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

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

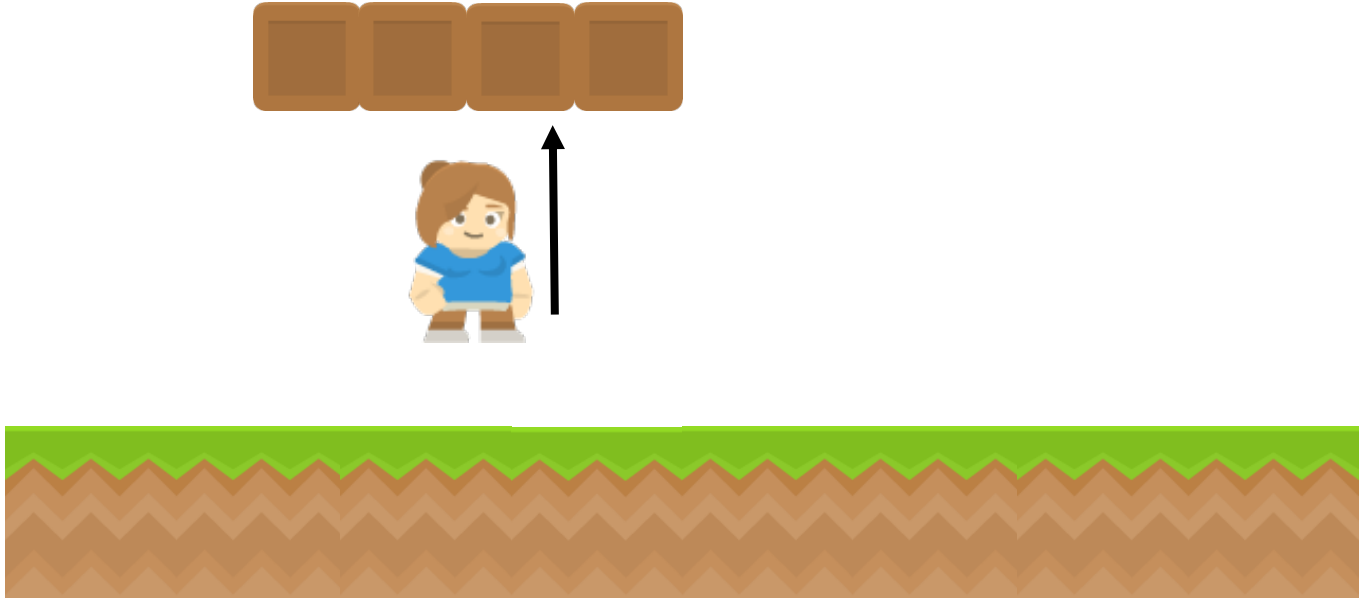
Resolving Platform Collisions



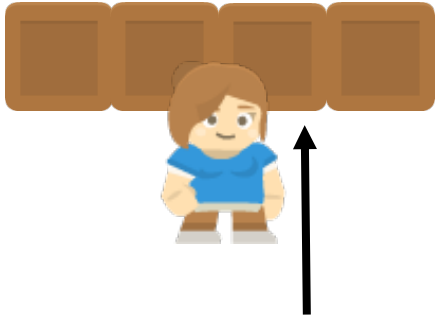
```
change_y += gravity
# move in vertical direction
center_y += change_y
# resolve collisions
…
# move in horizontal direction
center_x += change_x
# resolve collisions
…
```

Resolving Platform Collisions

move in vertical direction

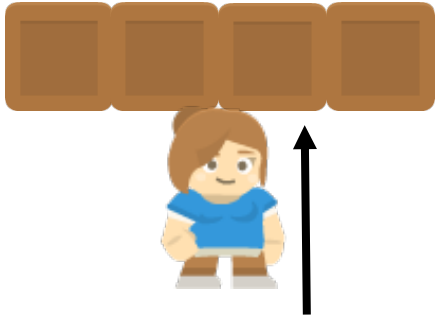


Resolving Platform Collisions



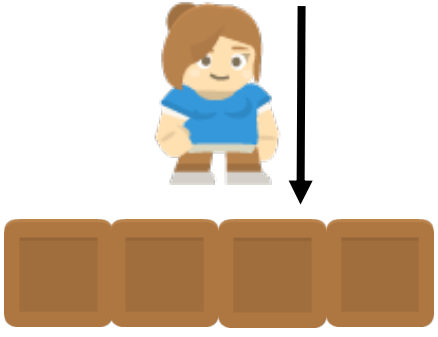
move in vertical direction
compute list of all platforms which collide with player
if list not empty:
    if player is moving up:

Resolving Platform Collisions



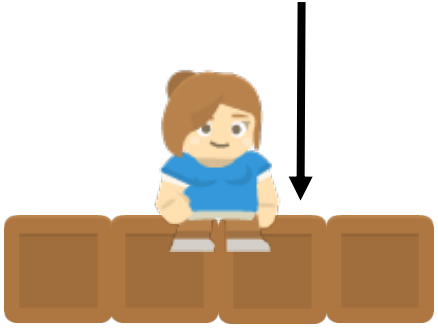
```
move in vertical direction
compute list of all platforms which collide with player
if list not empty:
    if player is moving up:
        set top of player = bottom of a collided platform
```

Resolving Platform Collisions



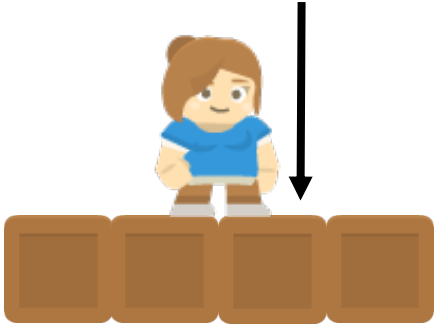
```
move in vertical direction
compute list of all platforms which collide with player
if list not empty:
    if player is moving up:
        set top of player = bottom of a collided platform
    if player is moving down:
```

Resolving Platform Collisions



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

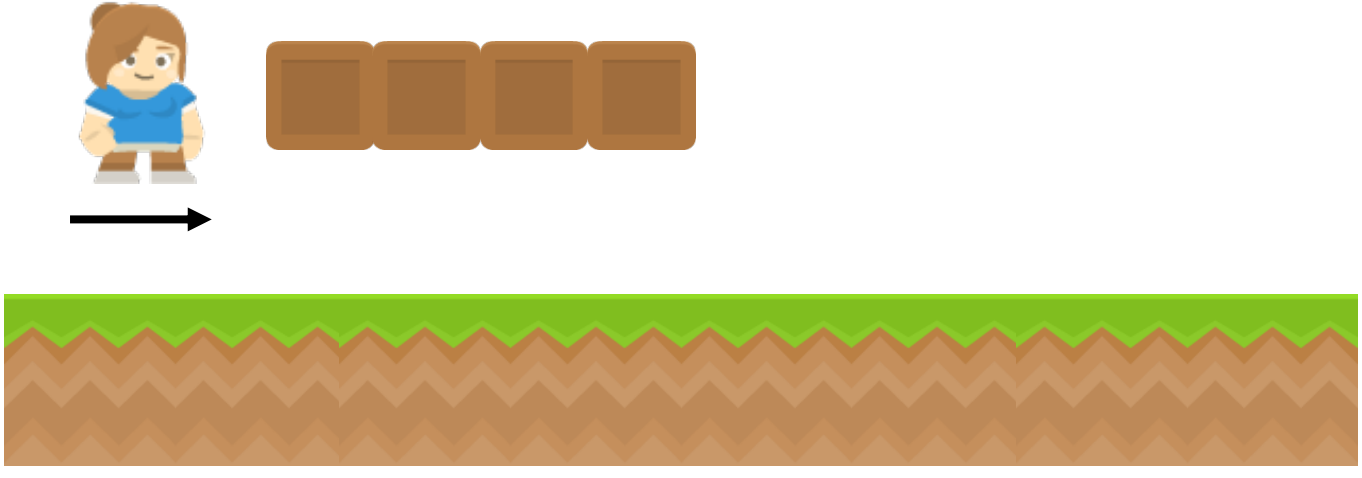
Resolving Platform Collisions



```
move in vertical direction
compute list of all platforms which collide with player
if list not empty:
    if player is moving up:
        set top of player = bottom of a collided platform
    if player is moving down:
        set bottom of player = top of a collided platform
    set player's change_y = 0
```

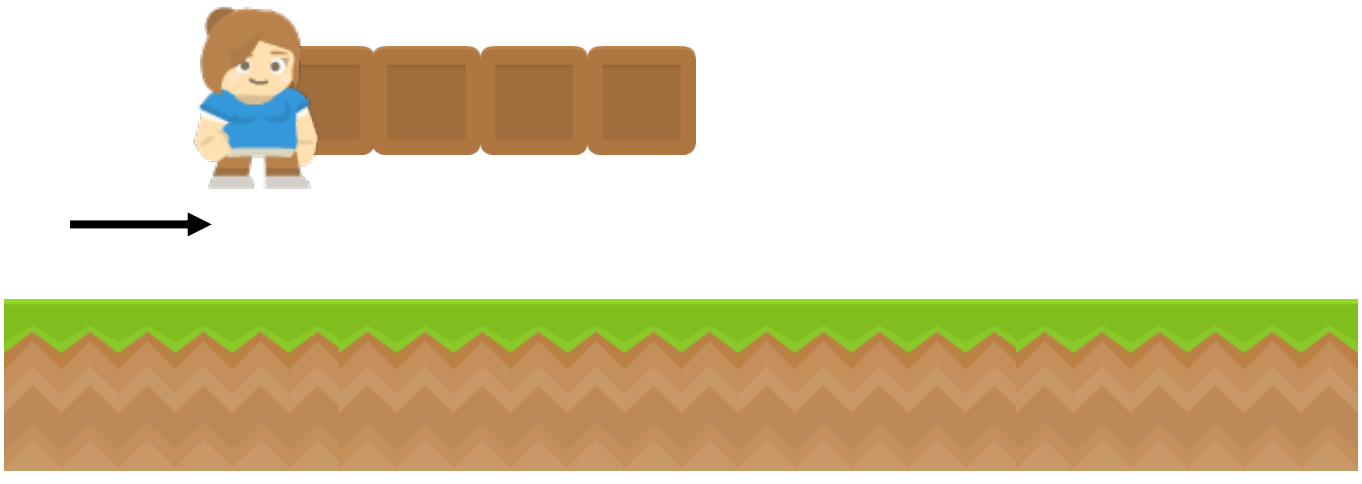
Resolving Platform Collisions

move in horizontal direction



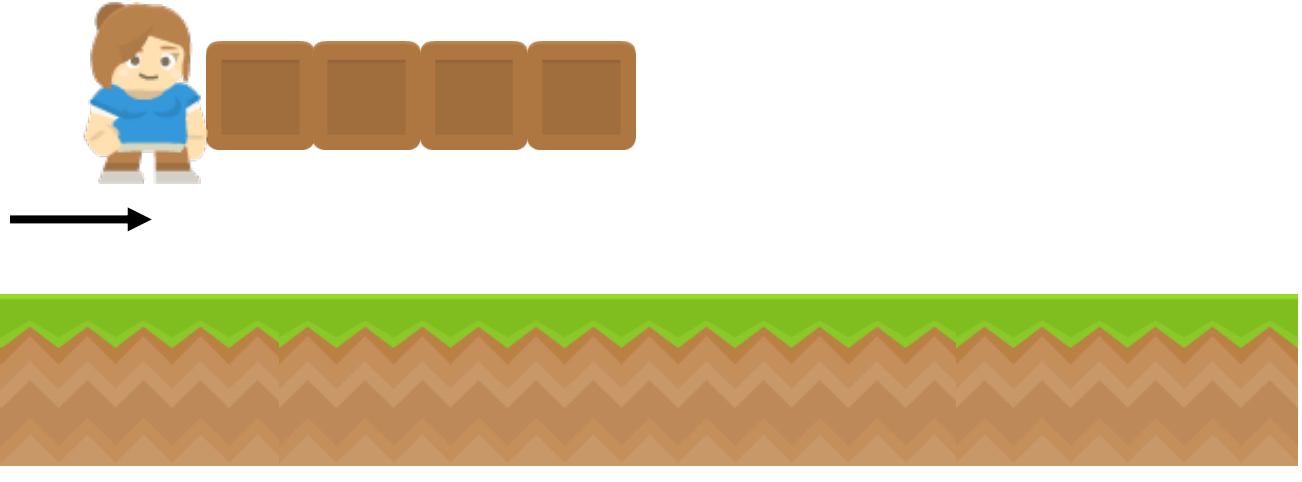
Resolving Platform Collisions

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



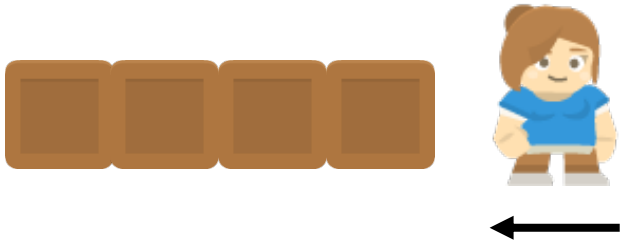
Resolving Platform Collisions

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



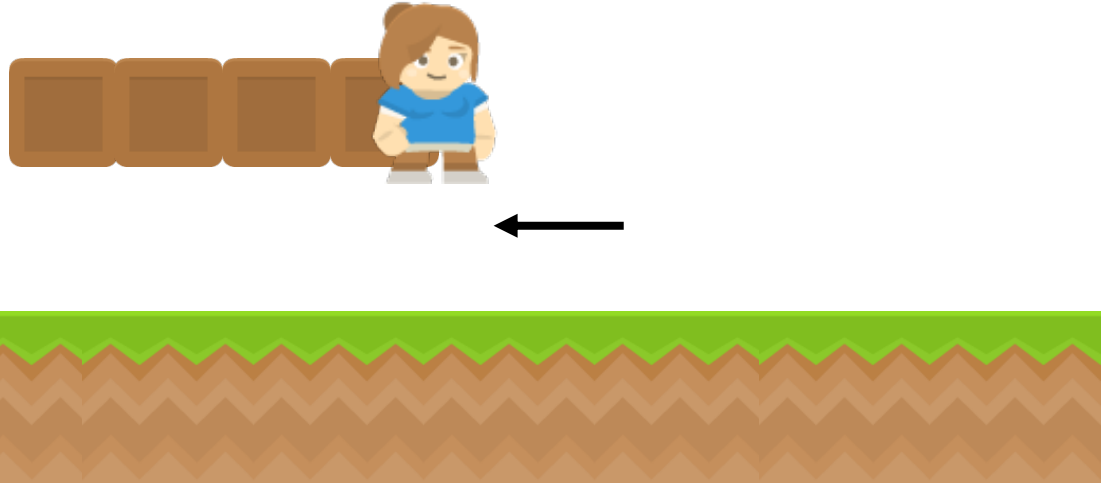
Resolving Platform Collisions

```
move in horizontal direction
compute list of all platforms which collide with player
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:
```



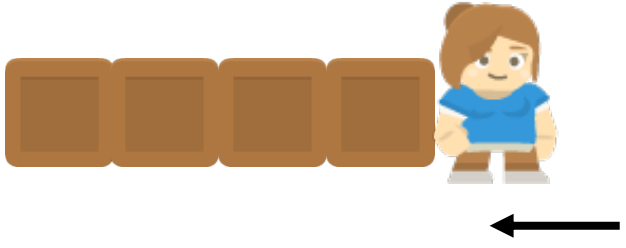
Resolving Platform Collisions

move in horizontal direction
compute list of all platforms which collide with player
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:



Resolving Platform Collisions

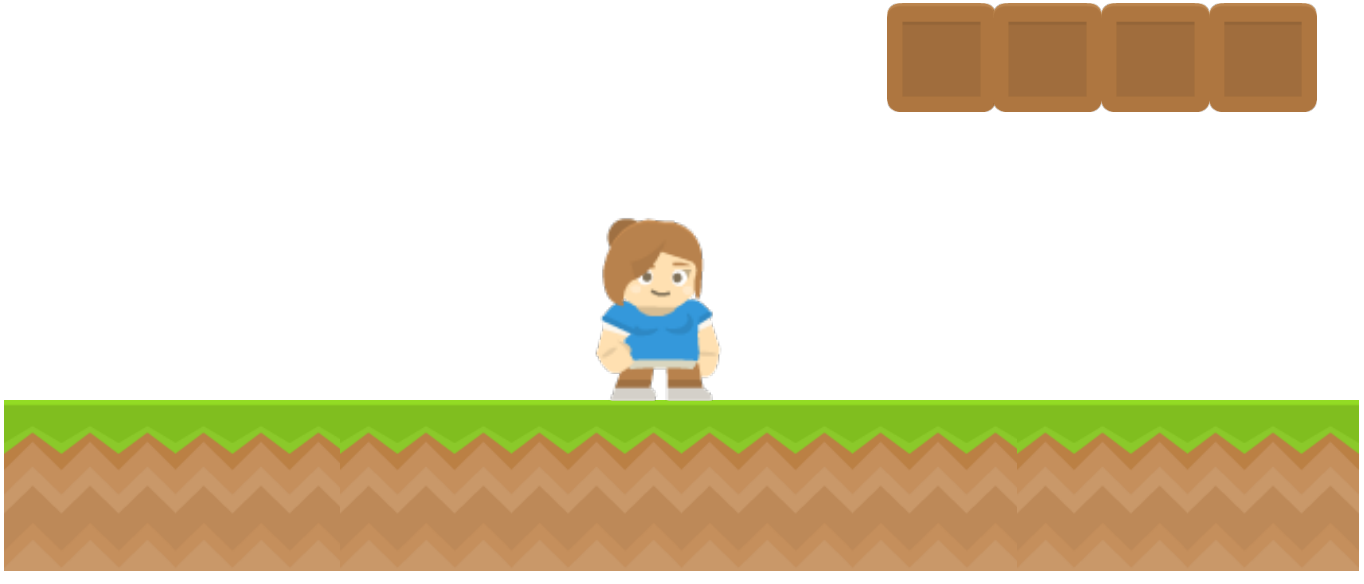
```
move in horizontal direction
compute list of all platforms which collide with player
if list not empty:
    if player is moving right:
        set right side of player = left side of a collided platform
    if player is moving left:
        set left side of player = right side of a collided platform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



Player Jumps

Jumping Rule: Player can only jump when he is on a platform.

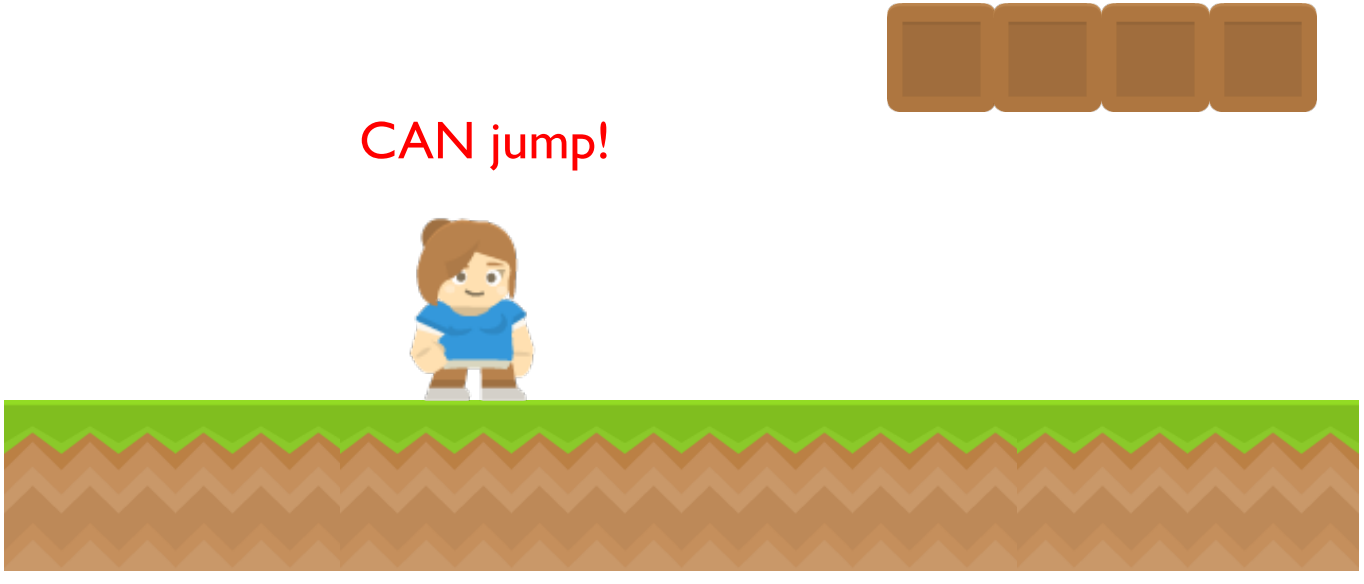
- No multi-jumping



Player Jumps

Jumping Rule: Player can only jump when he is on a platform.

- No multi-jumping



Player Jumps

Jumping Rule: Player can only jump when he is on a platform.

- No multi-jumping

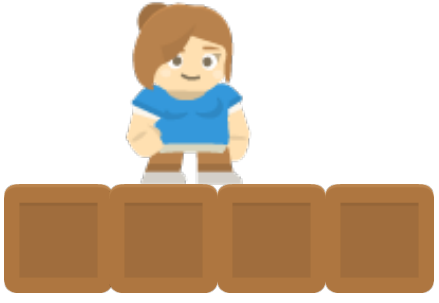
CANNOT jump!

is_on_platform(sprite, platforms)

This method returns whether the sprite is on one of the platforms.

Algorithm:

move sprite down say 5 pixels



is_on_platform(sprite, platforms)

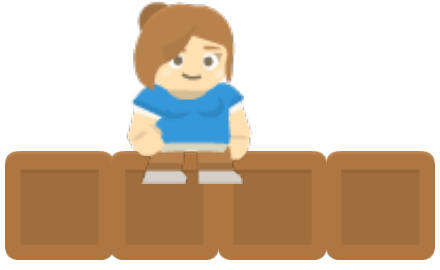
This method returns whether the sprite is on one of the platforms.

Algorithm:

move sprite down say 5 pixels

compute collision list with platforms

restore position by moving up 5 pixels



is_on_platform(sprite, platforms)

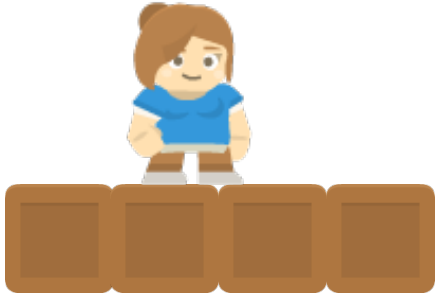
This method returns whether the sprite is on one of the platforms.

Algorithm:

move sprite down say 5 pixels

compute collision list with platforms

restore position by moving up 5 pixels



is_on_platform(sprite, platforms)

This method returns whether the sprite is on one of the platforms.

Algorithm:

move sprite down say 5 pixels

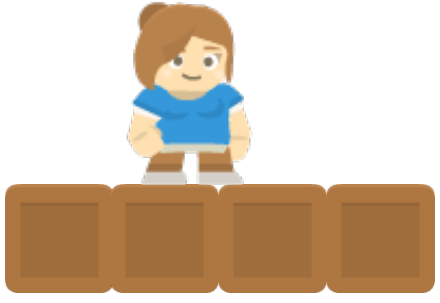
compute collision list with platforms

restore position by moving up 5 pixels

if collision list not empty

    return true

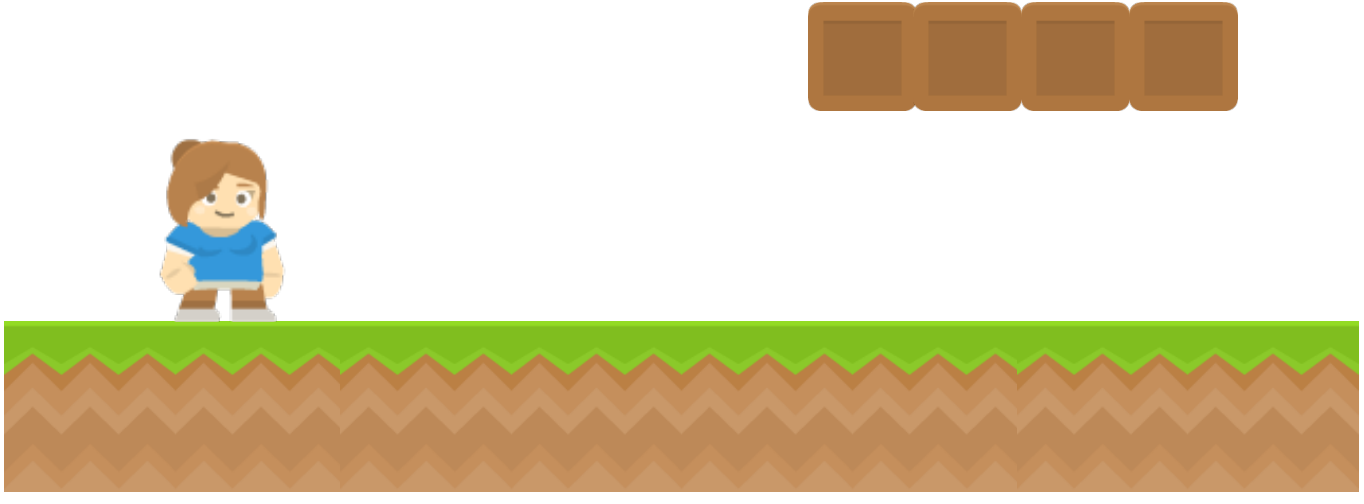
otherwise return false



Jumps

if key pressed is A and sprite is on platform:

    sprite.change_y = -JUMP_SPEED



Jumps

if key pressed is A and sprite is on platform:

    sprite.change_y = -JUMP_SPEED