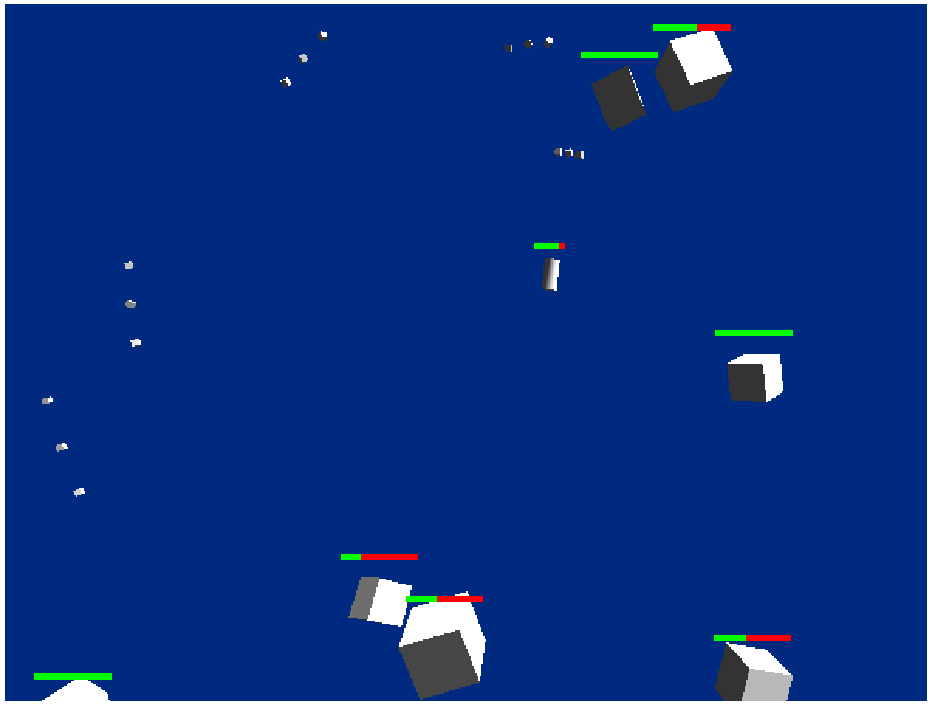
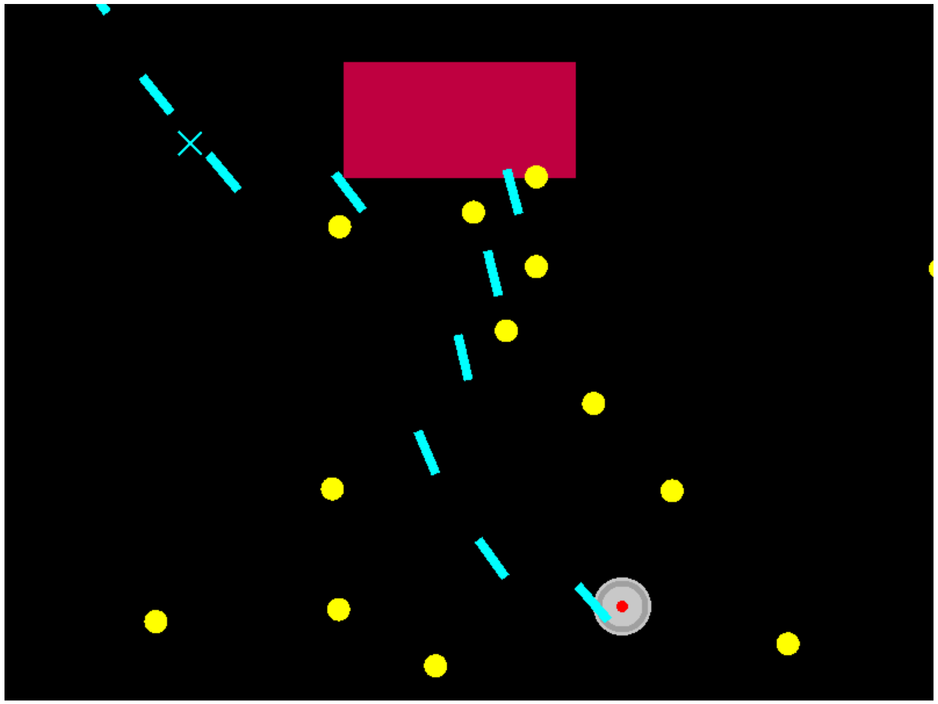
Low-Level Game Development

Prototype Games

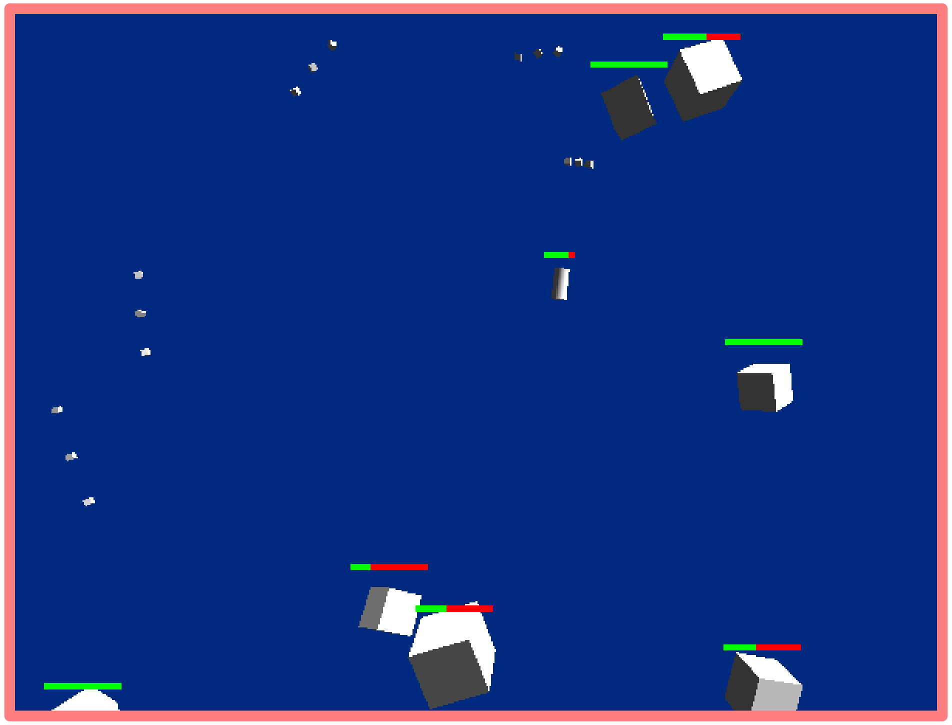
High Level



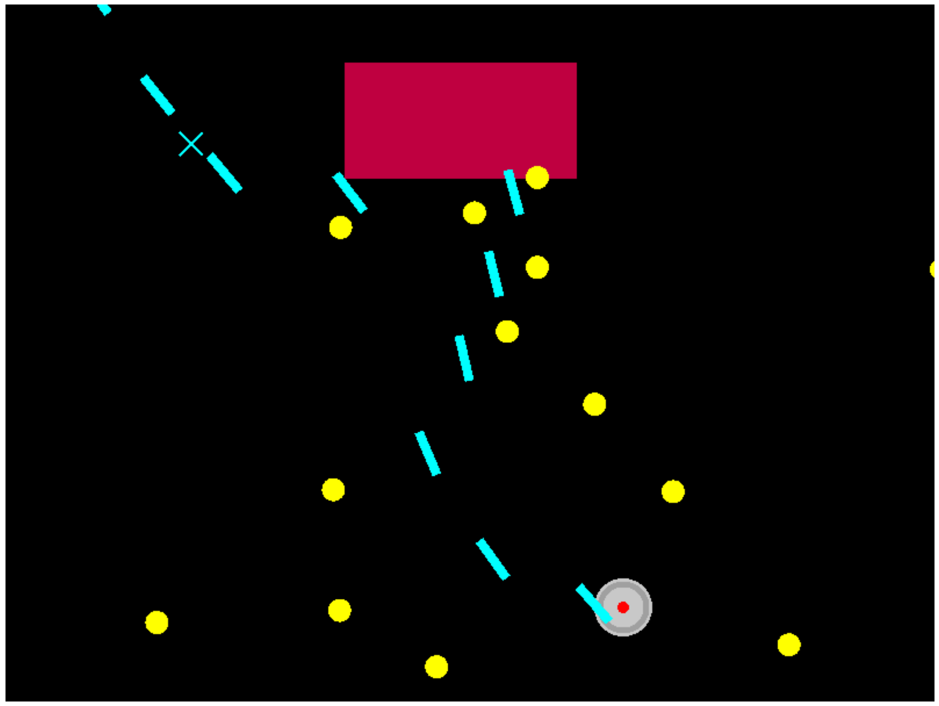
Low Level

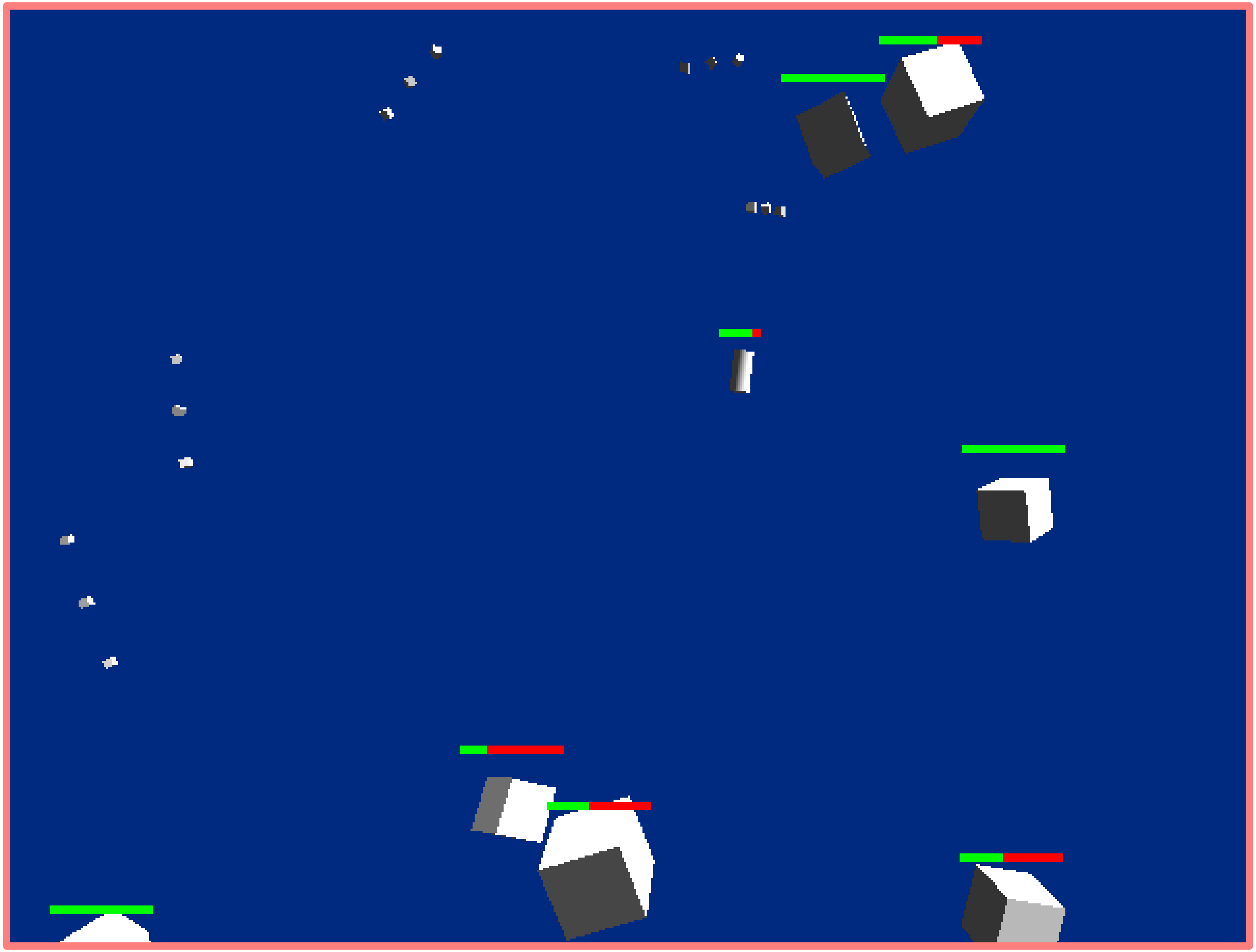


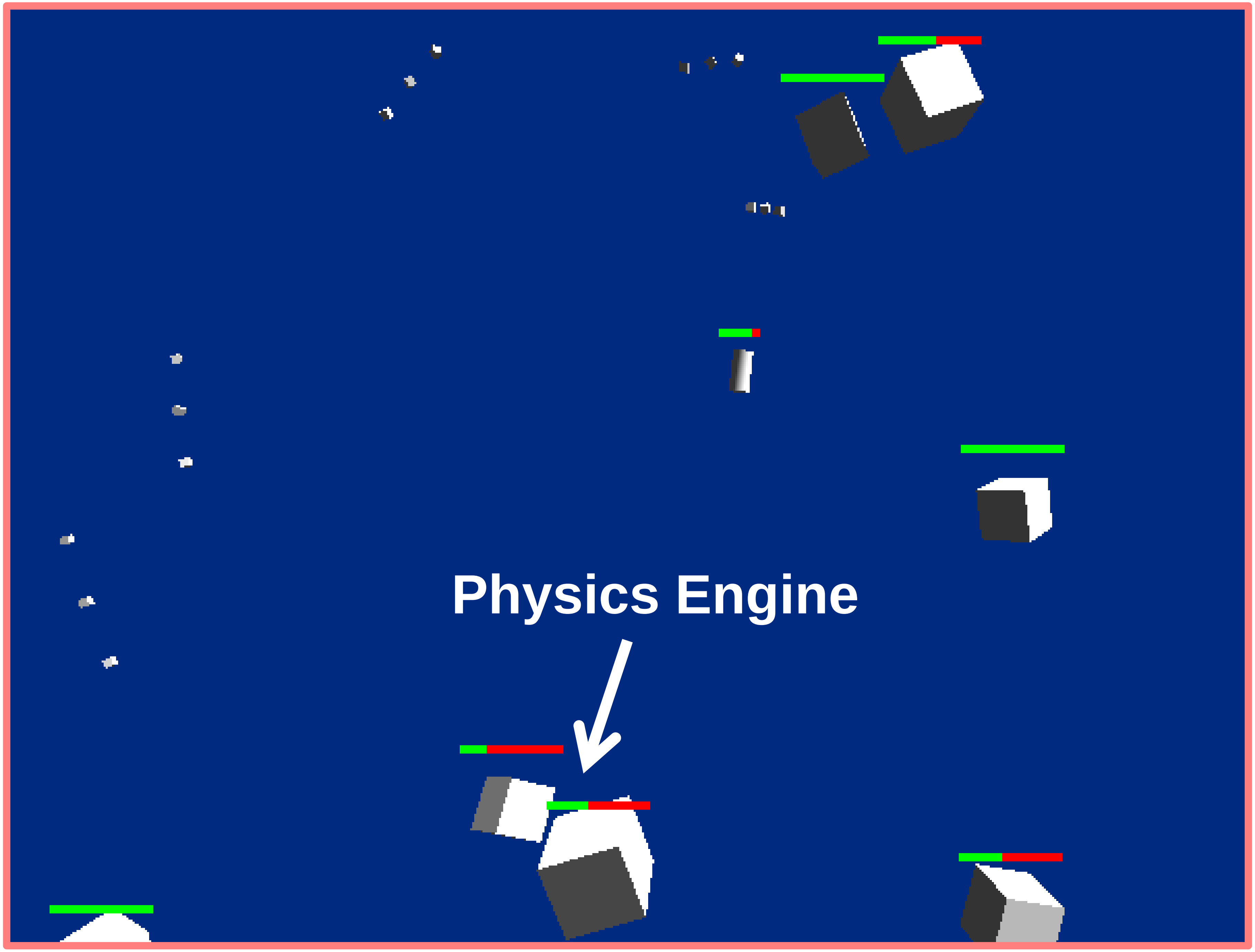
High Level

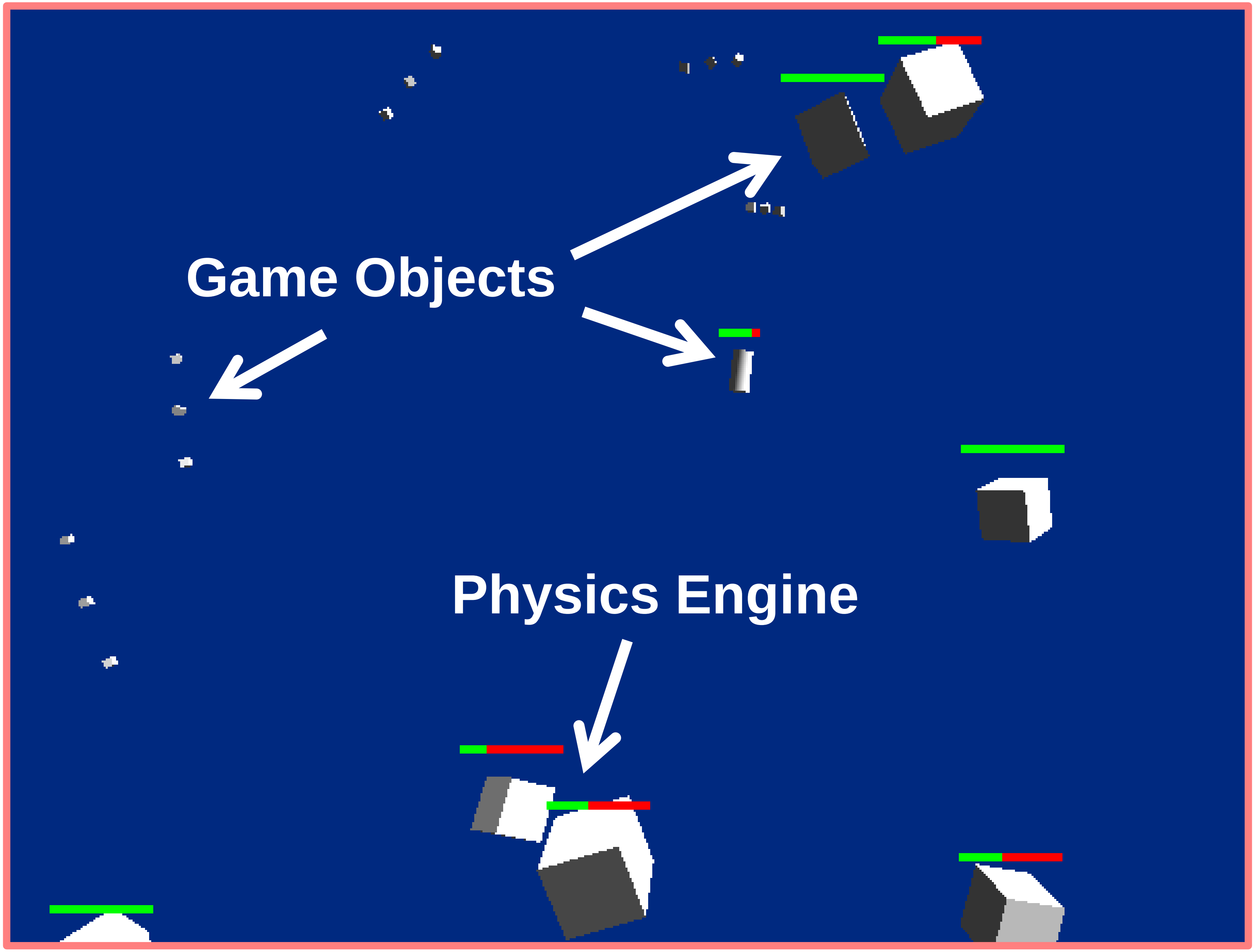


Low Level

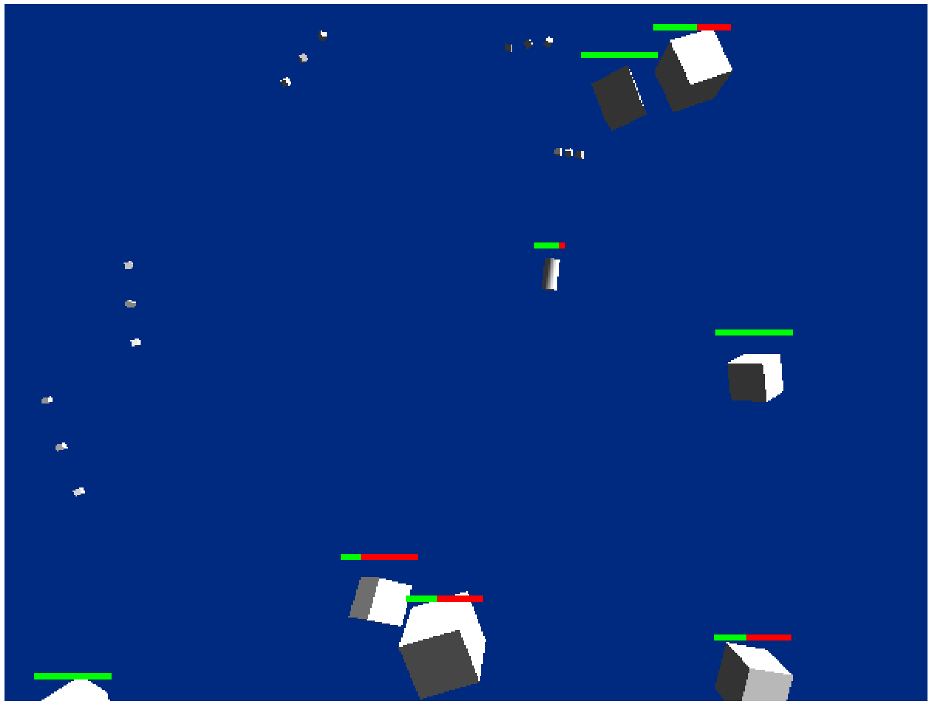




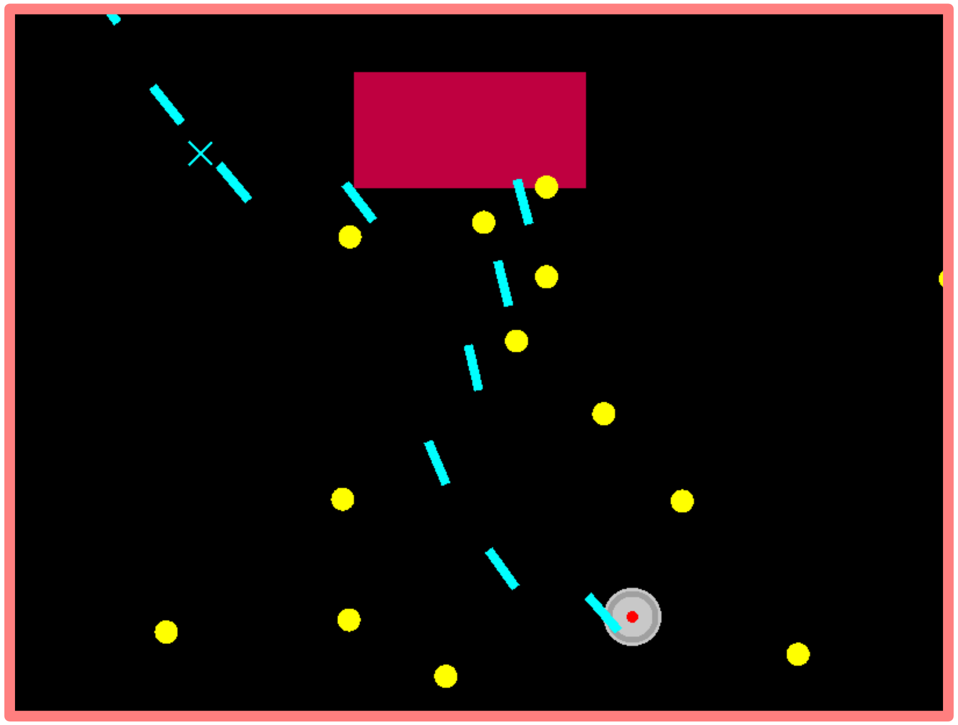

Physics Engine

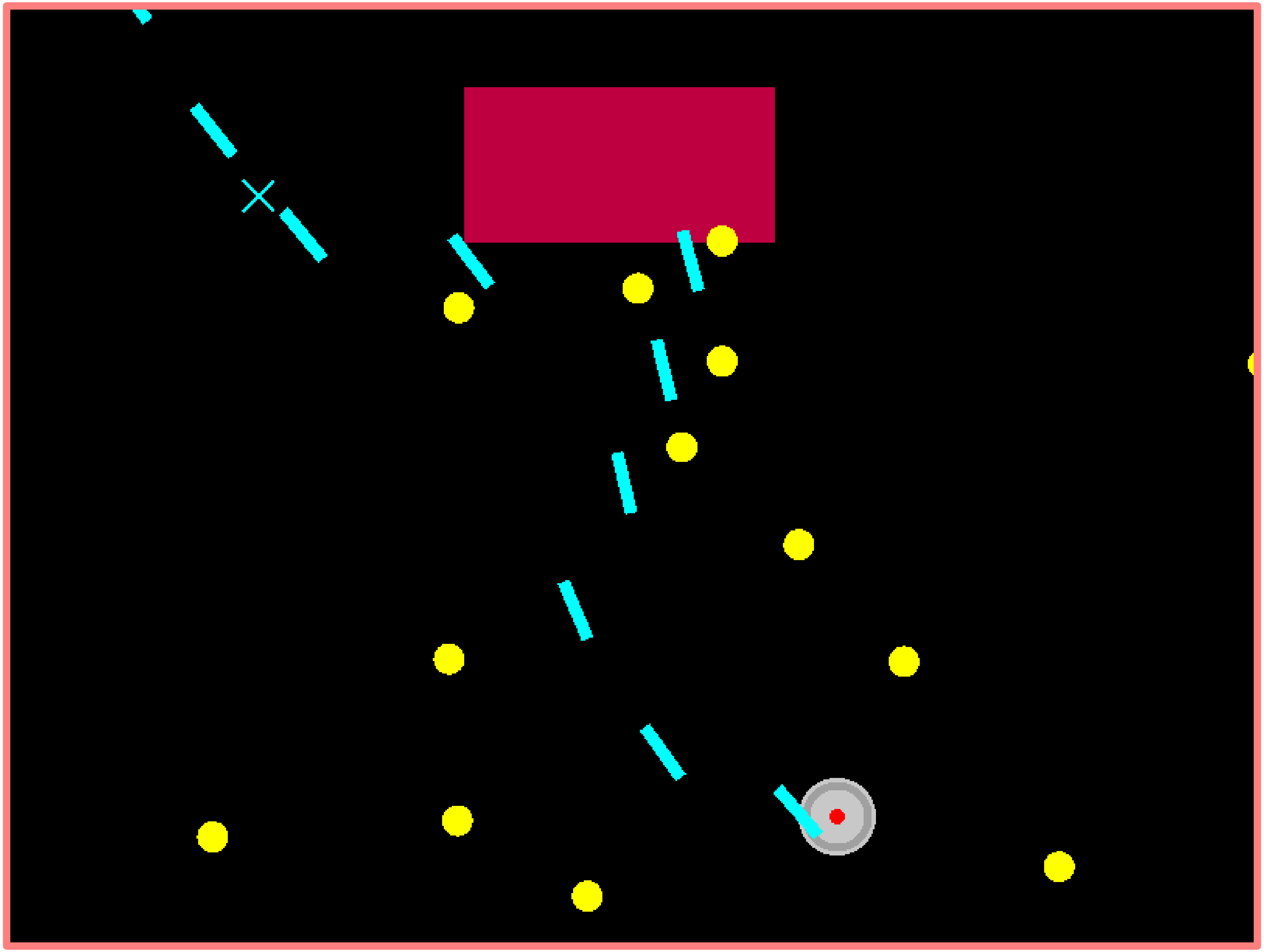

Game Objects
Physics Engine

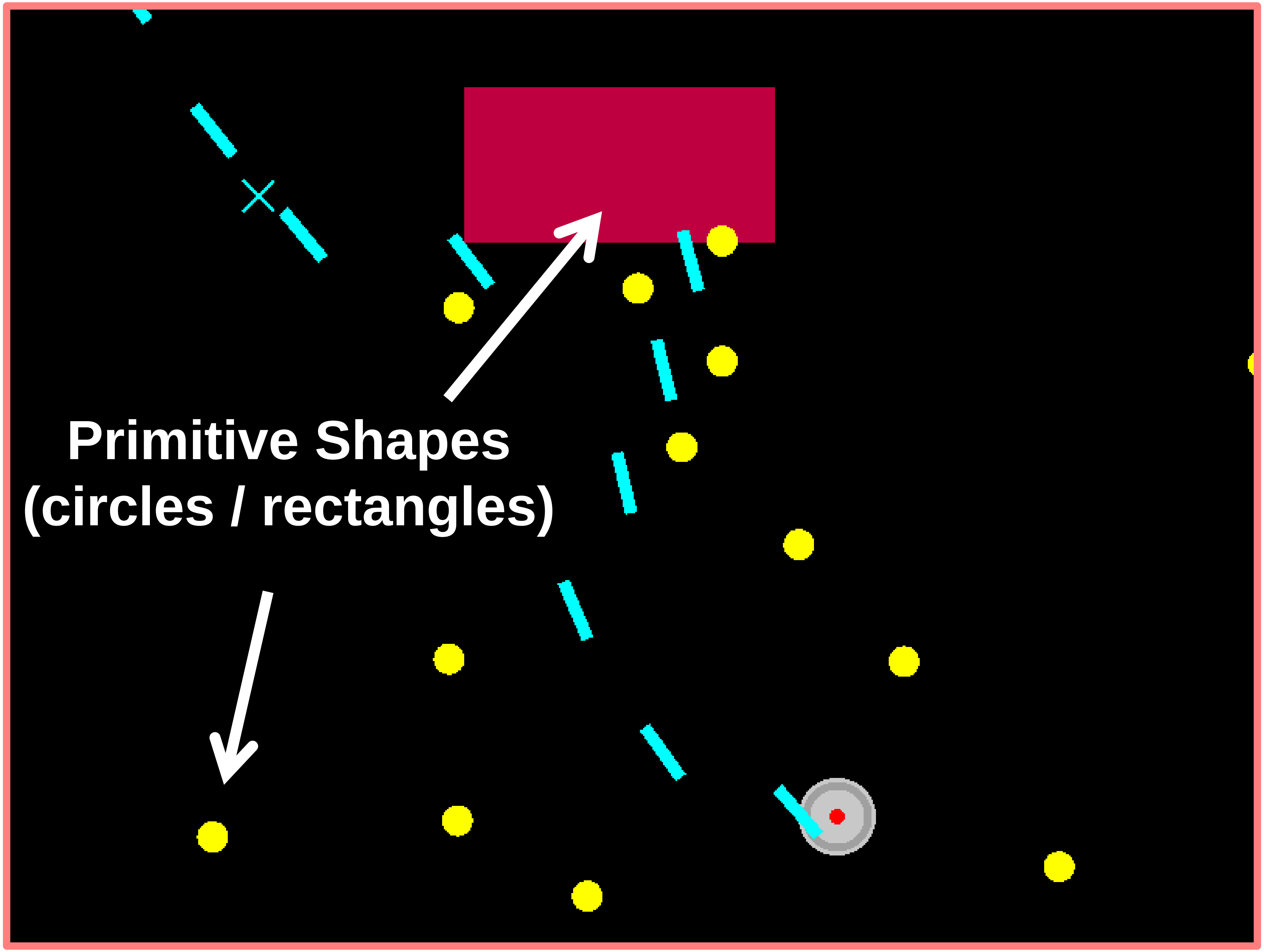
High Level

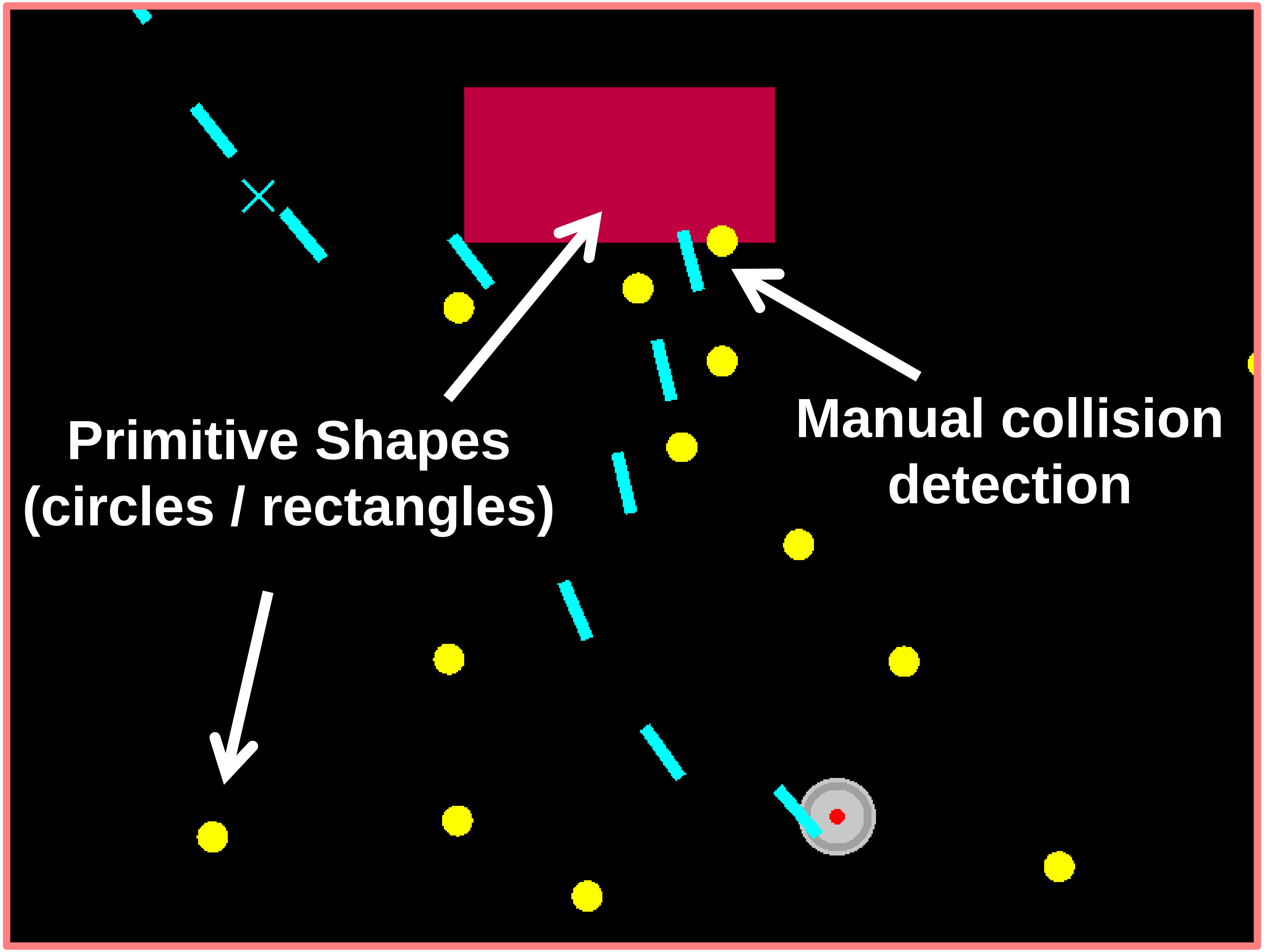


Low Level





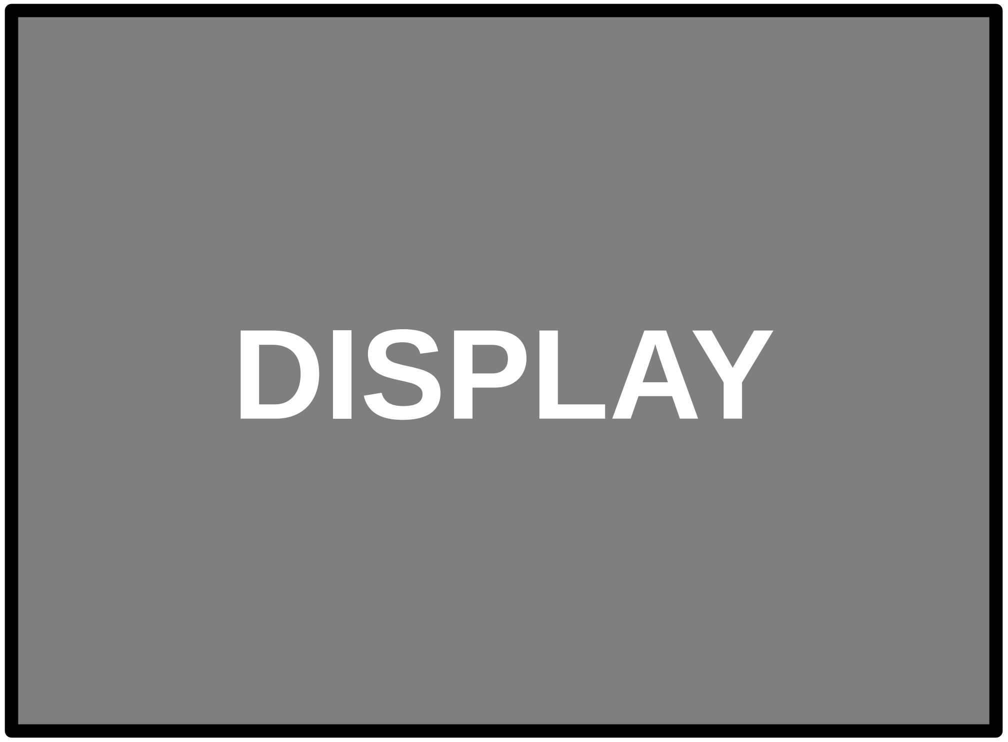

Primitive Shapes
(circles / rectangles)

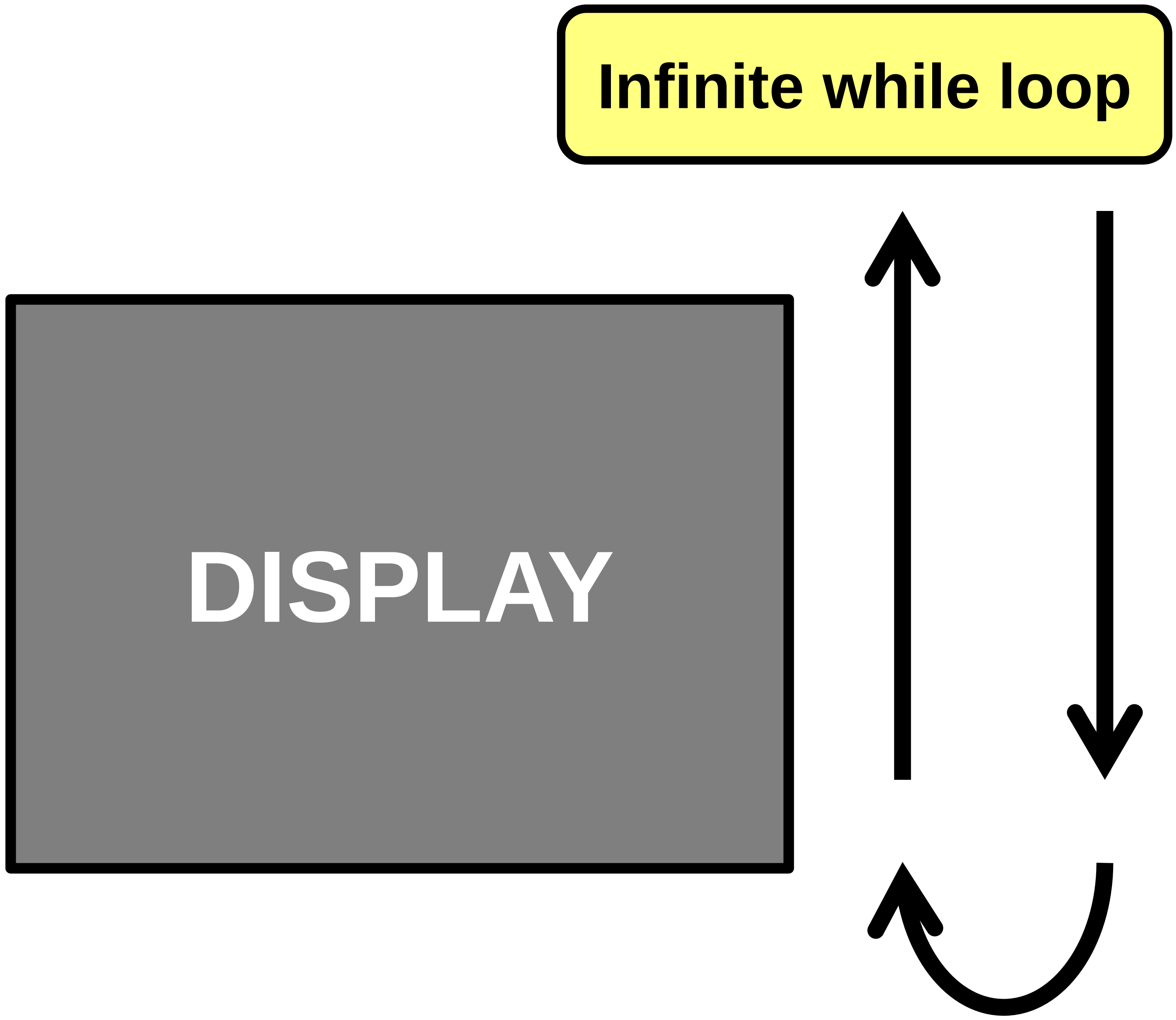

Primitive Shapes (circles / rectangles)
Manual collision detection

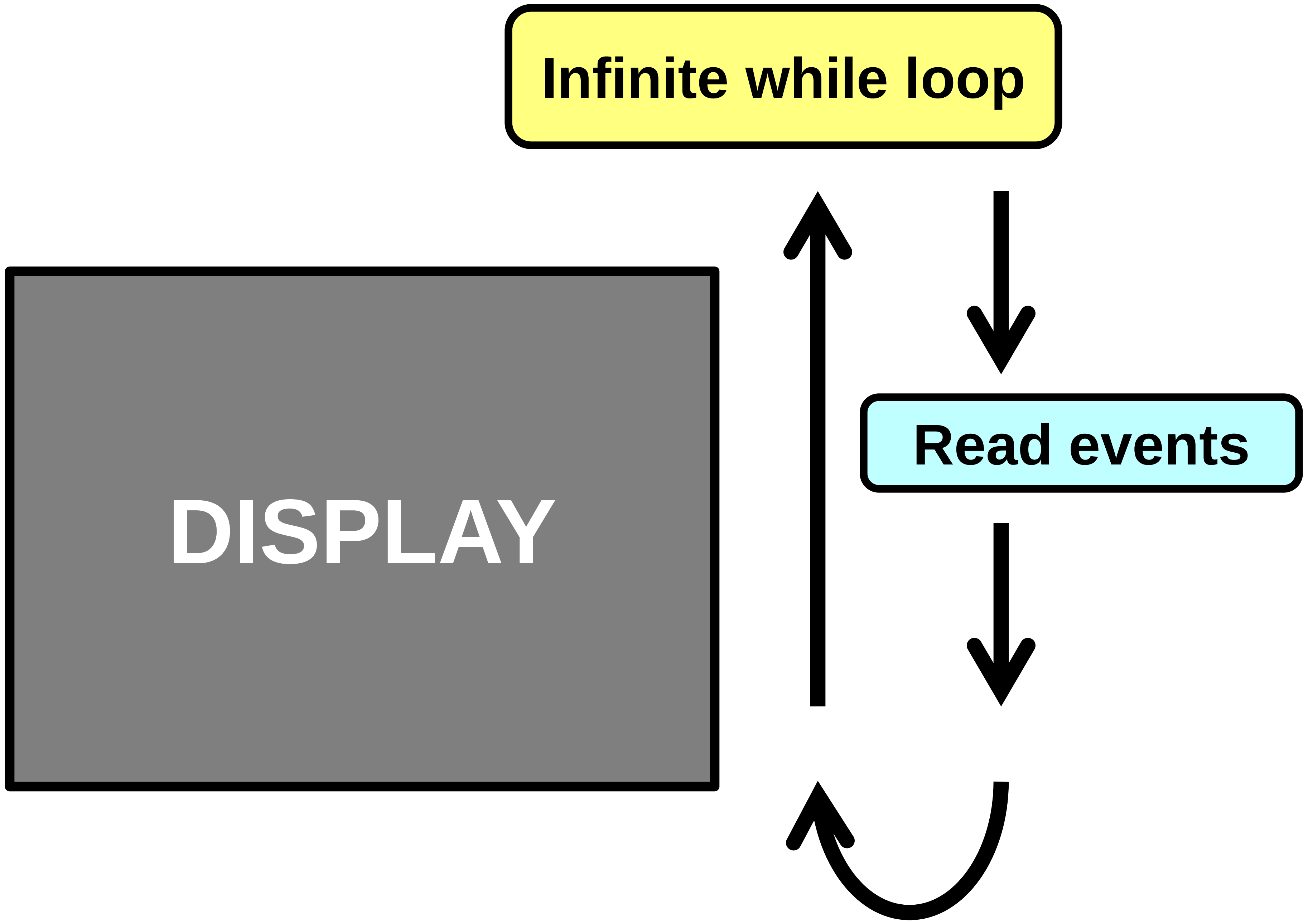
#includes
(for this workshop)

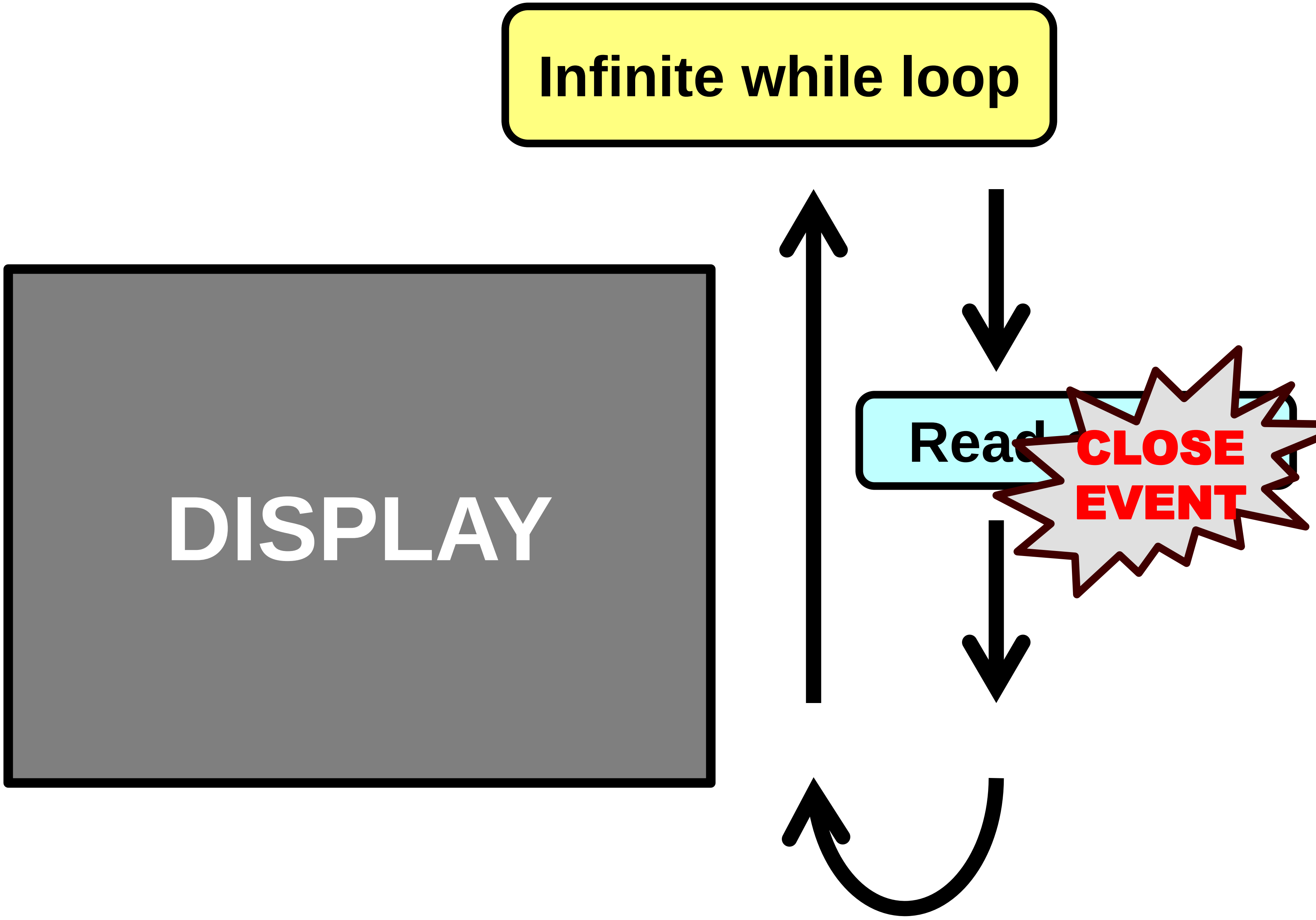
```
#include <SFML/Graphics.hpp>
#include <iostream>
#include <vector>
#include <cmath>
```

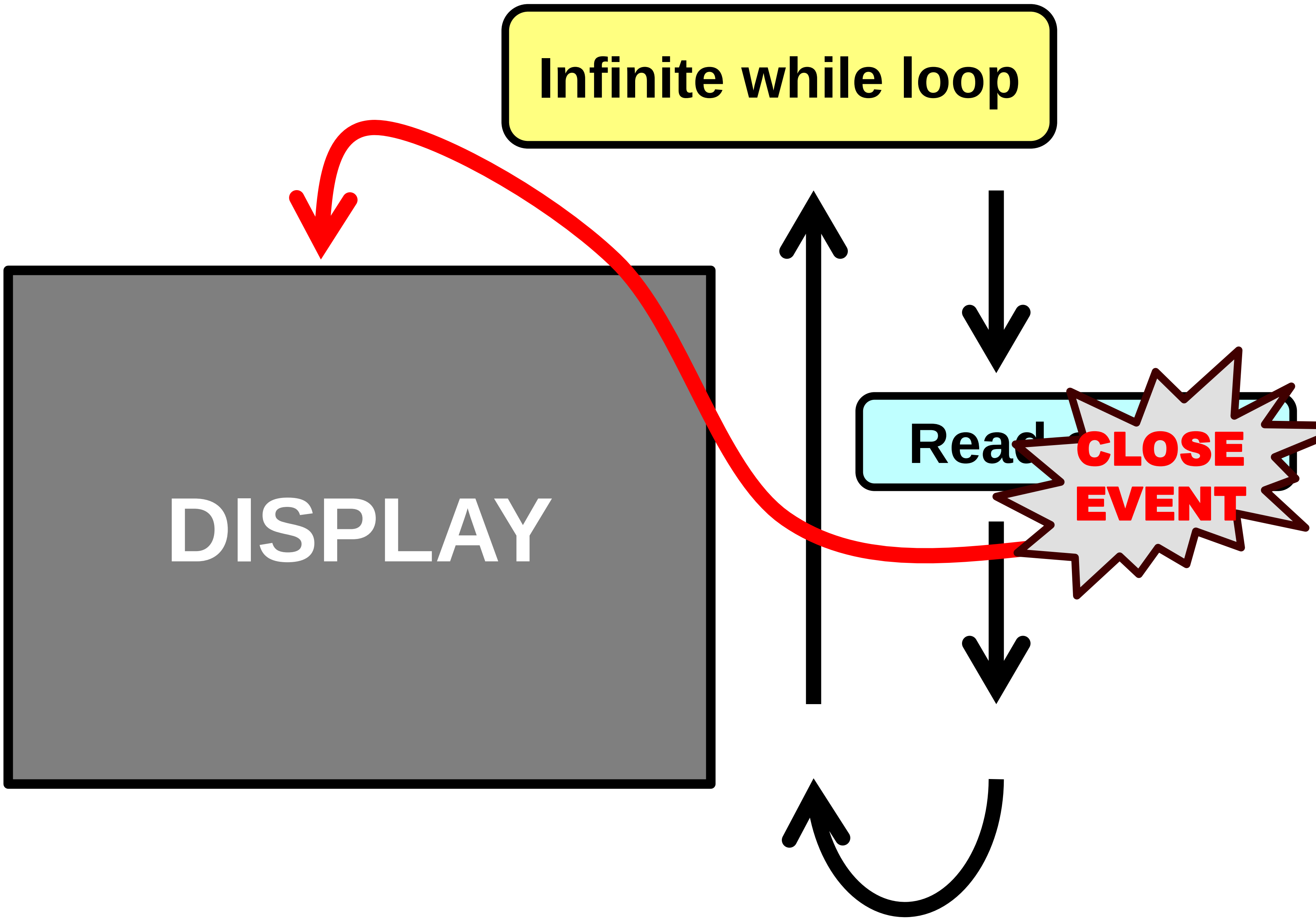
Display

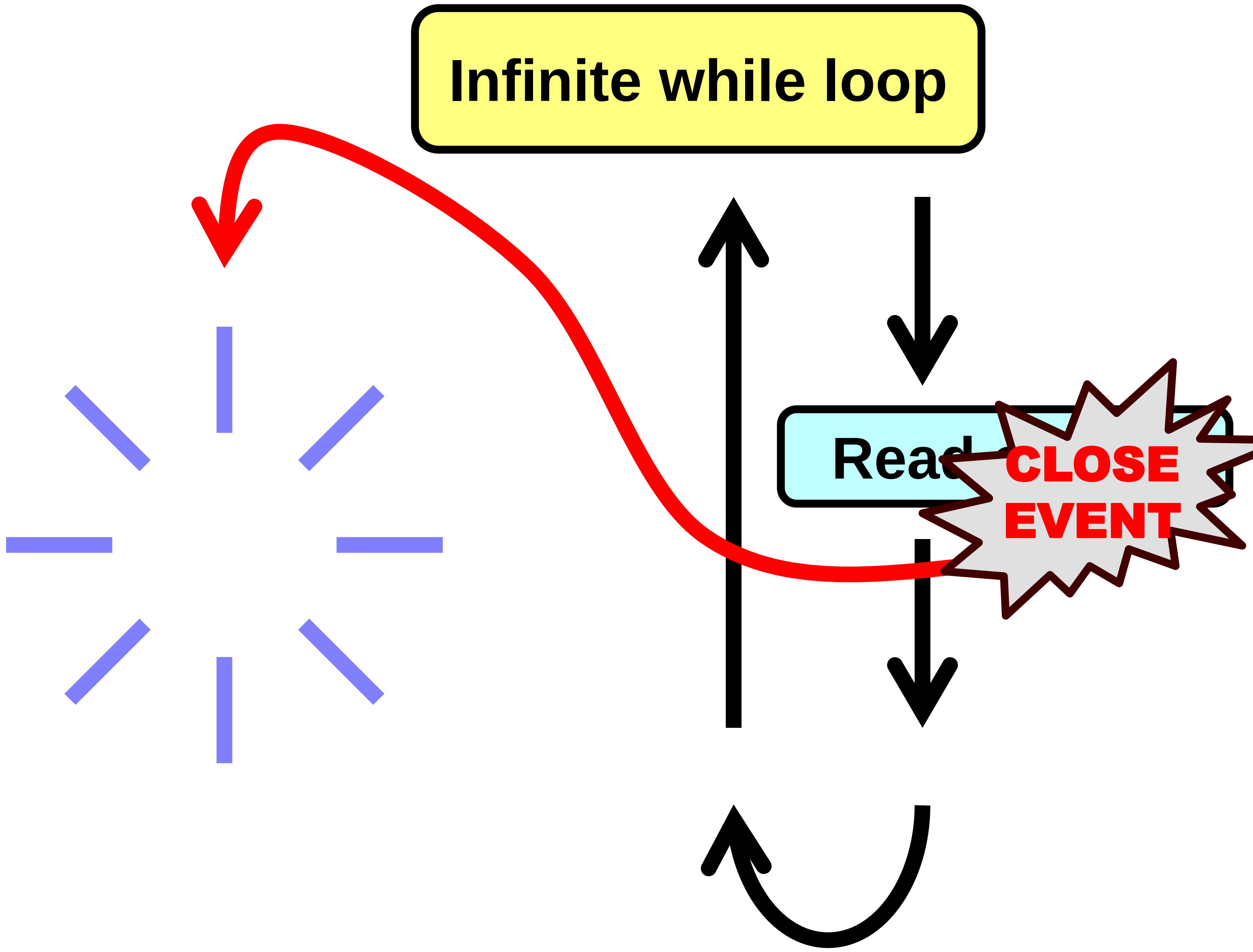

DISPLAY

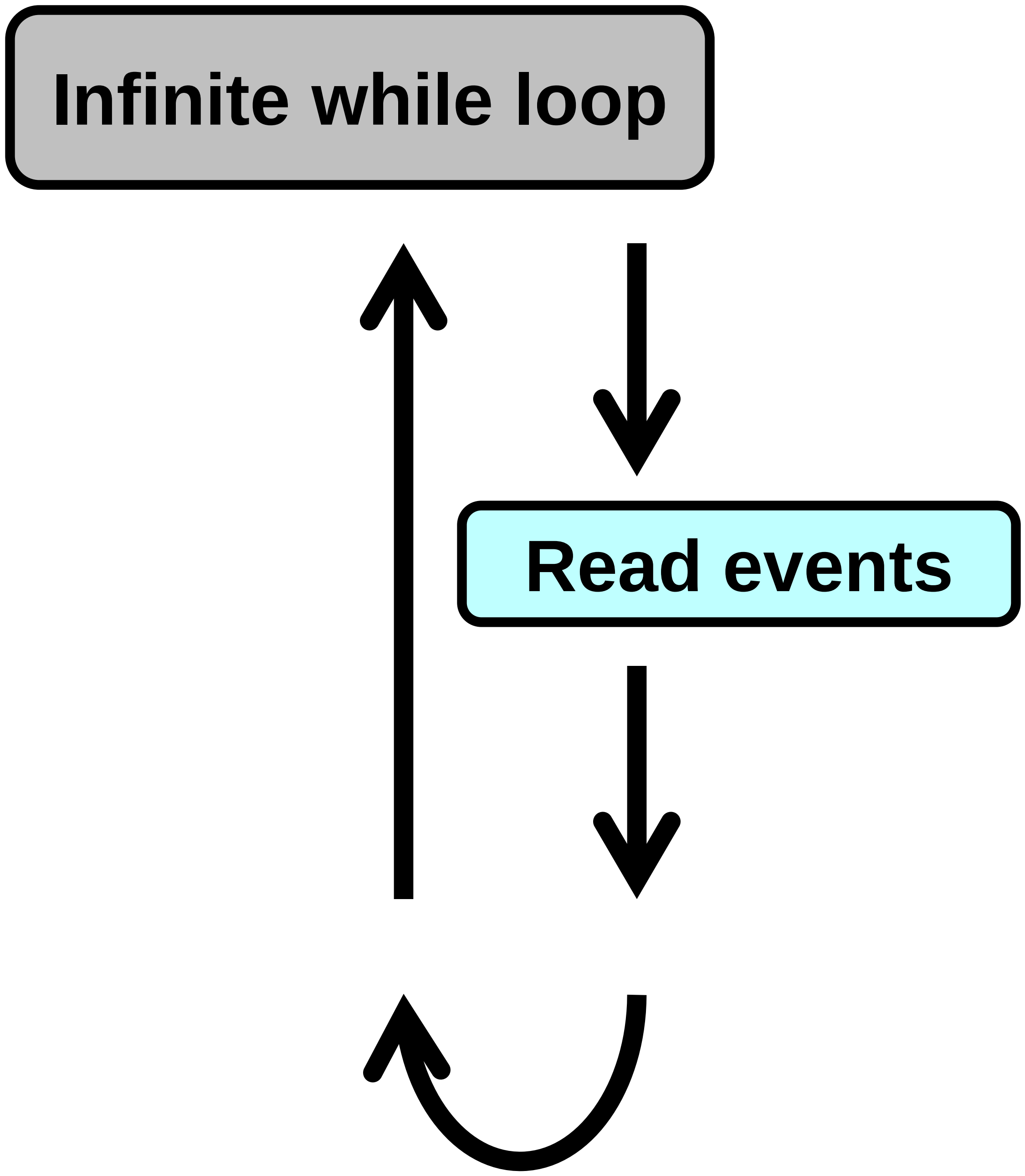

Infinite while loop
DISPLAY


Infinite while loop
Read events
DISPLAY


Infinite while loop
DISPLAY
Read
CLOSE EVENT

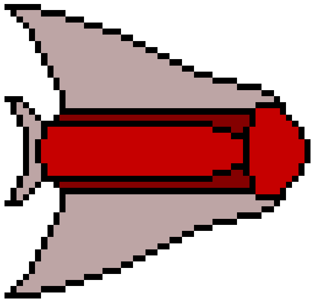

Infinite while loop
DISPLAY
Read
CLOSE EVENT


Infinite while loop
Read
CLOSE EVENT

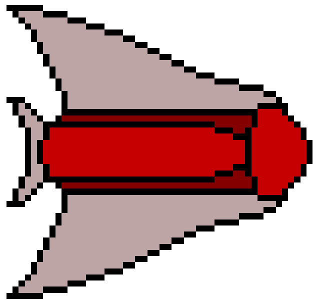

Infinite while loop
Read events

V0

Display



x = 3, y = 5



$x = 3, y = 5$

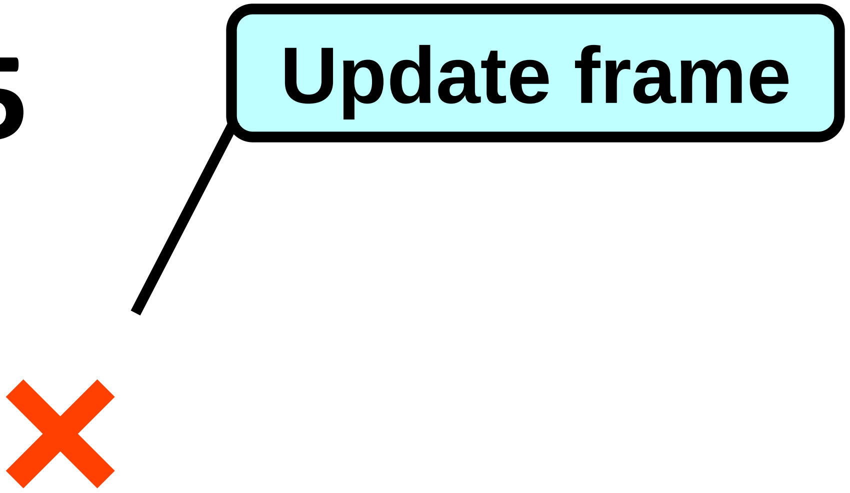
✗

Update Frame

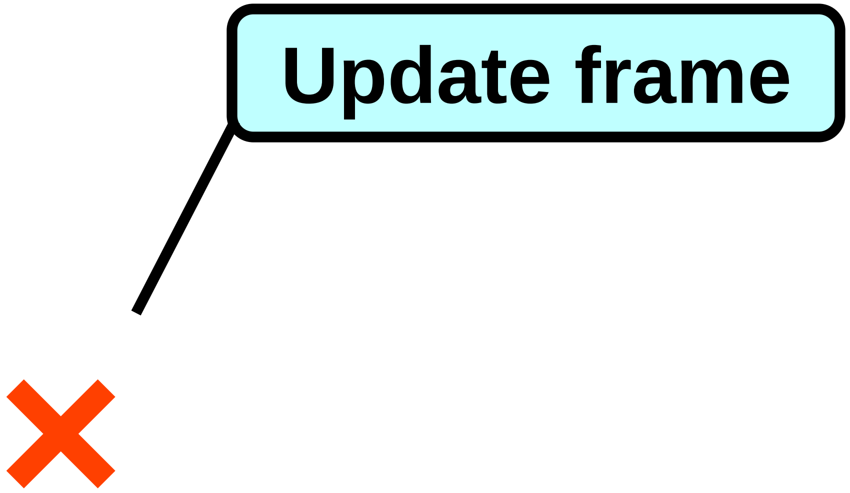
$x = 3, y = 5$

✗

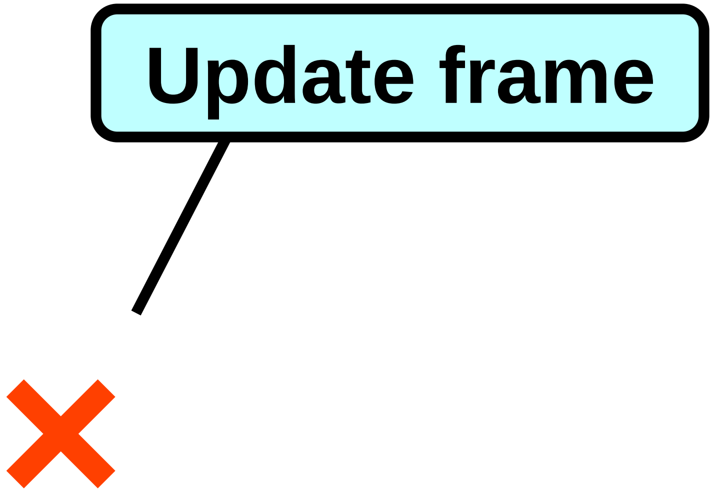
$x = 4, y = 5$



x = 5, y = 5



$x = 6, y = 5$

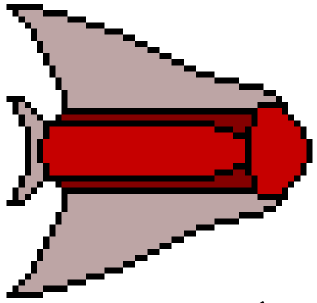


Draw Frame

$x = 3, y = 5$

✗

x = 3, y = 5

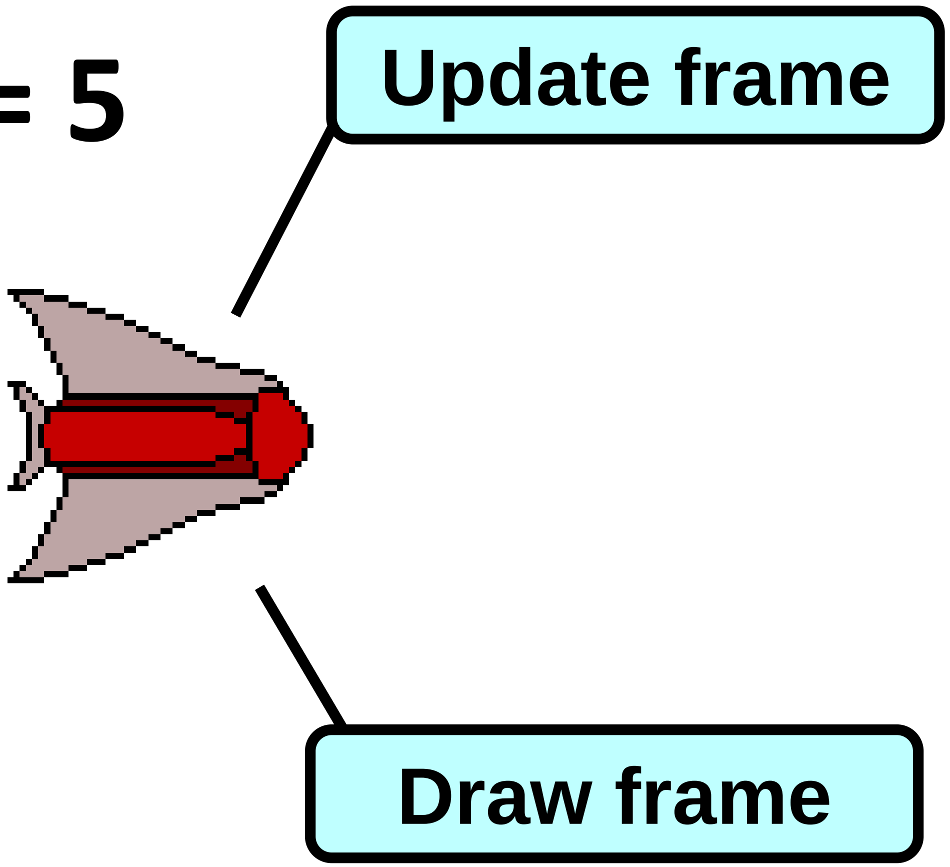


Draw frame

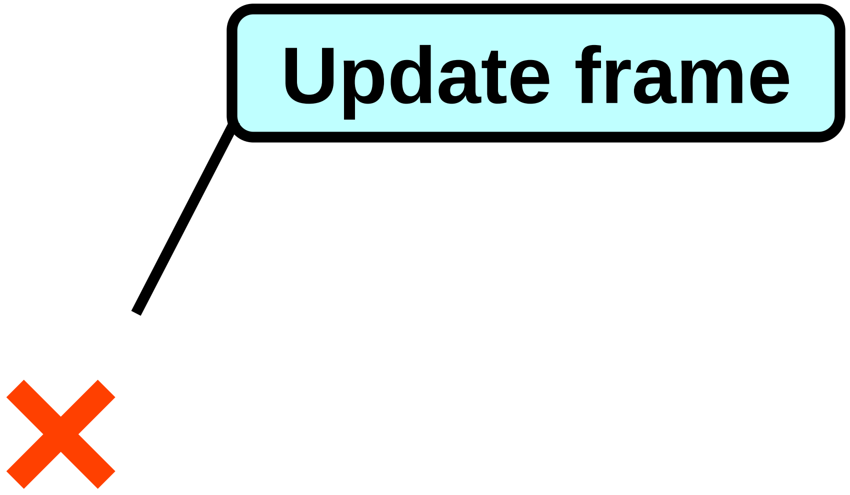
$x = 4, y = 5$

Update frame

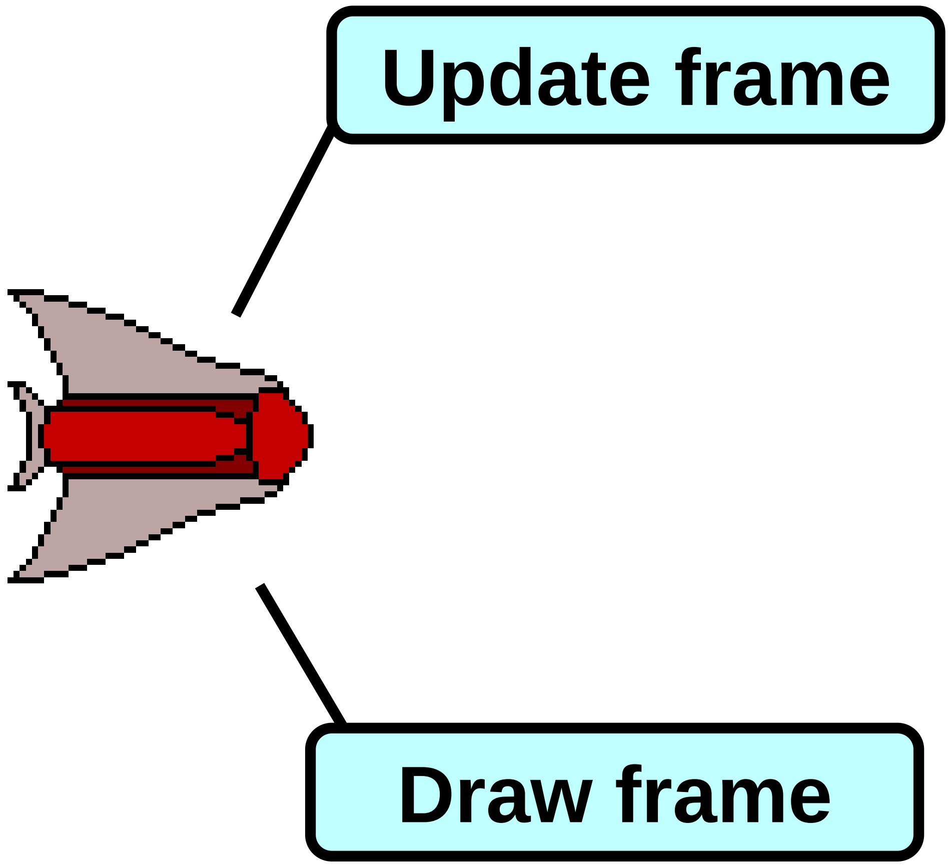
x = 4, y = 5



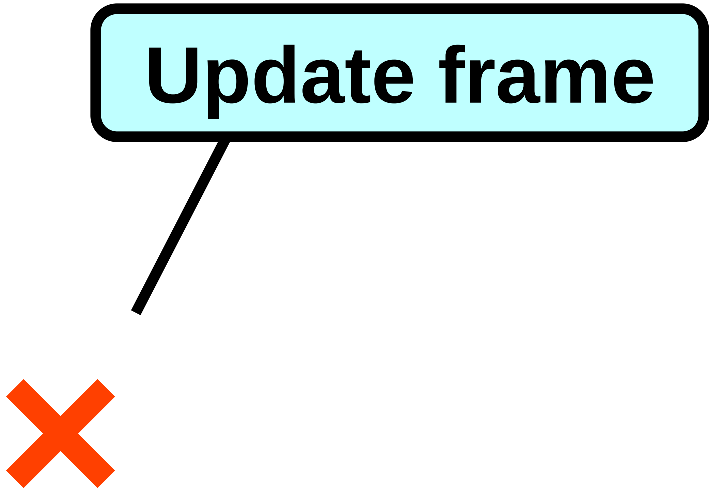
$x = 5, y = 5$



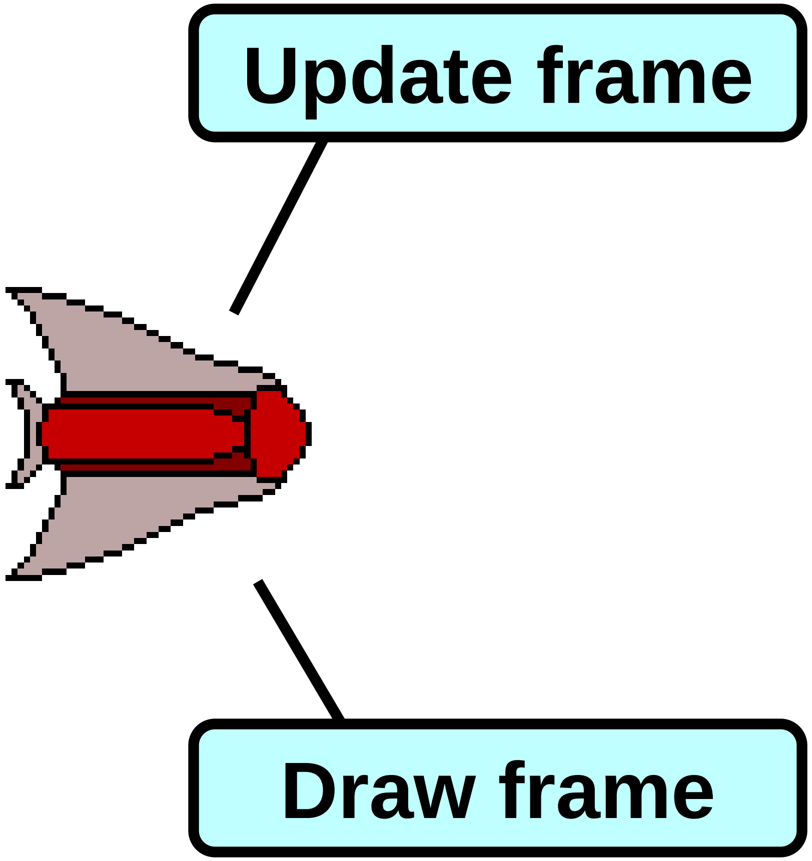
x = 5, y = 5



$x = 6, y = 5$



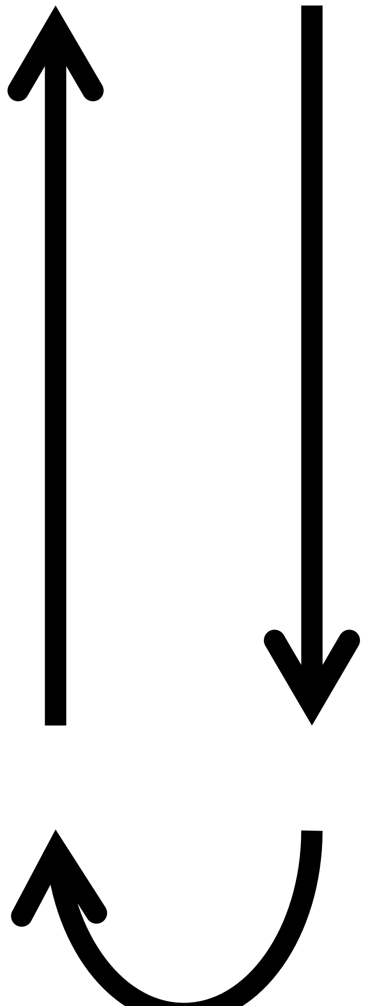
x = 6, y = 5

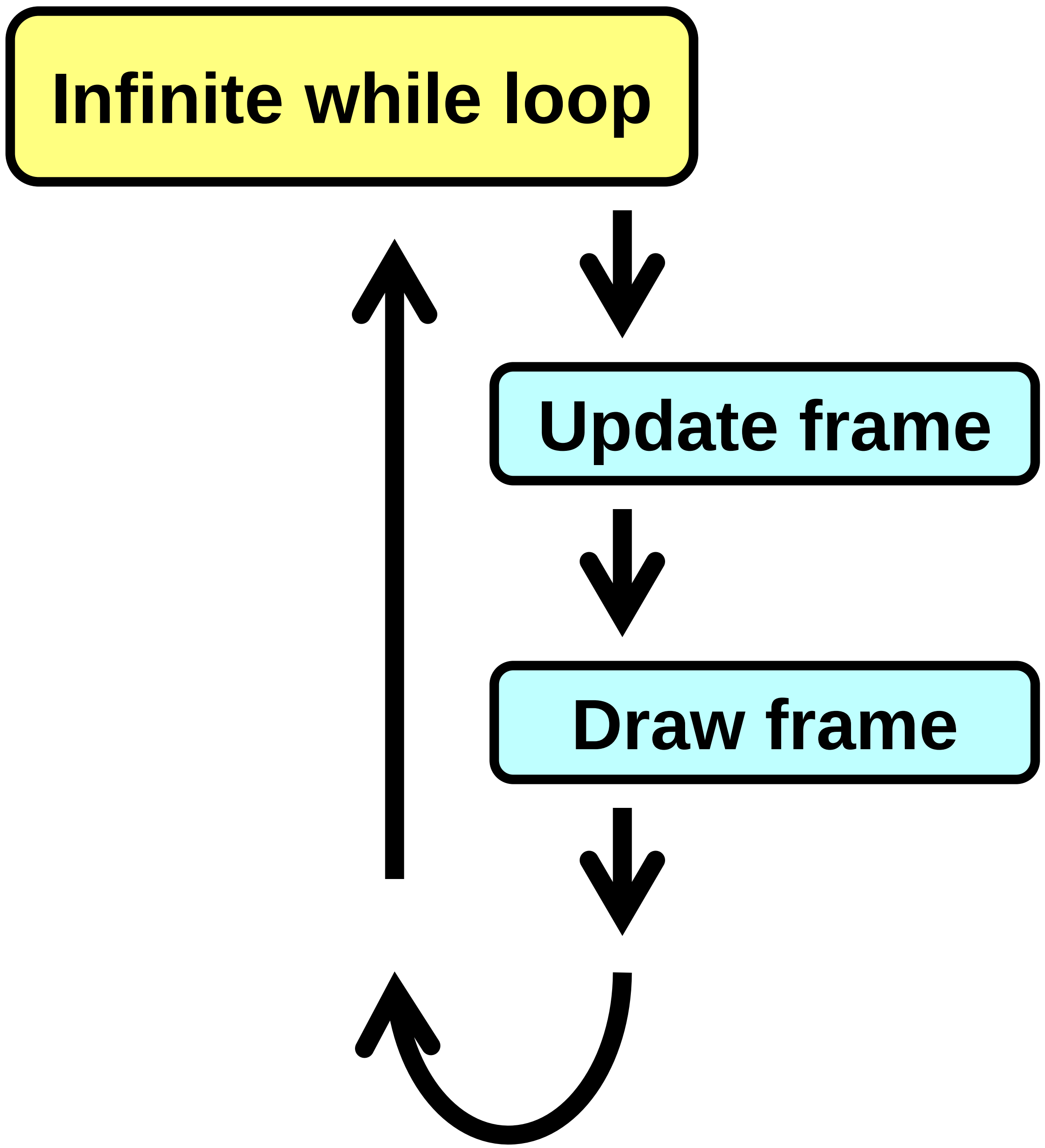


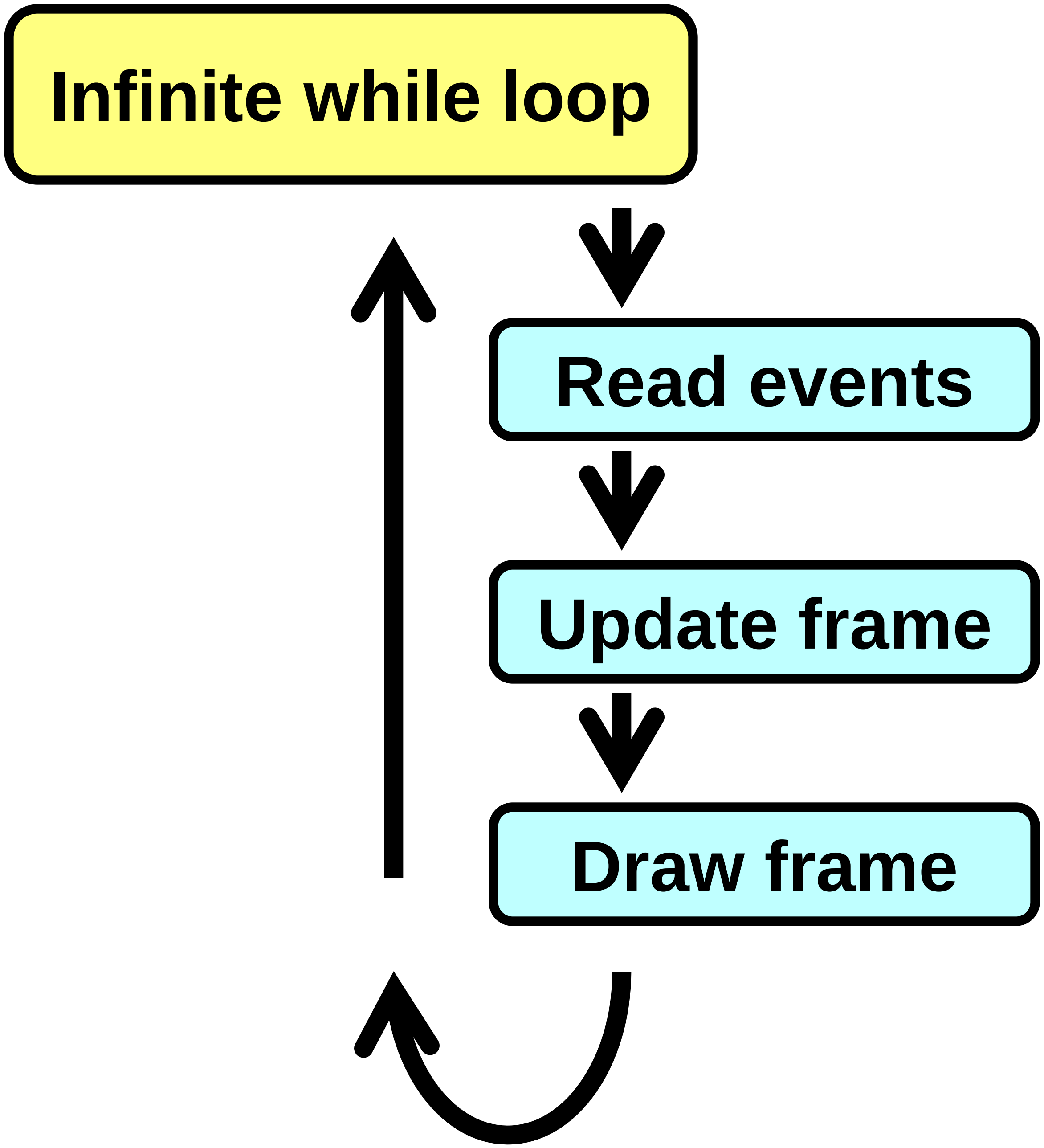
Structure

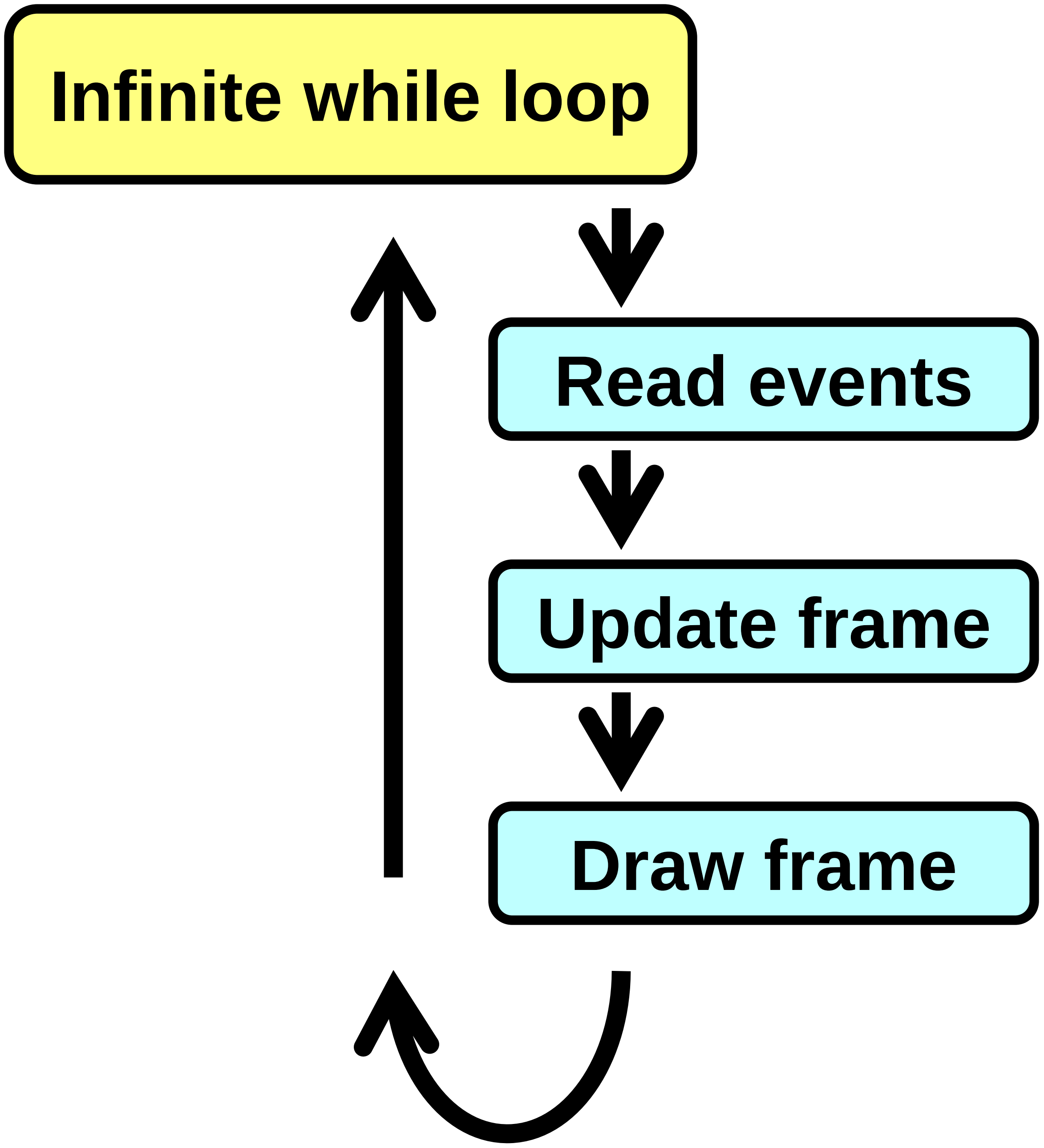
Infinite while loop

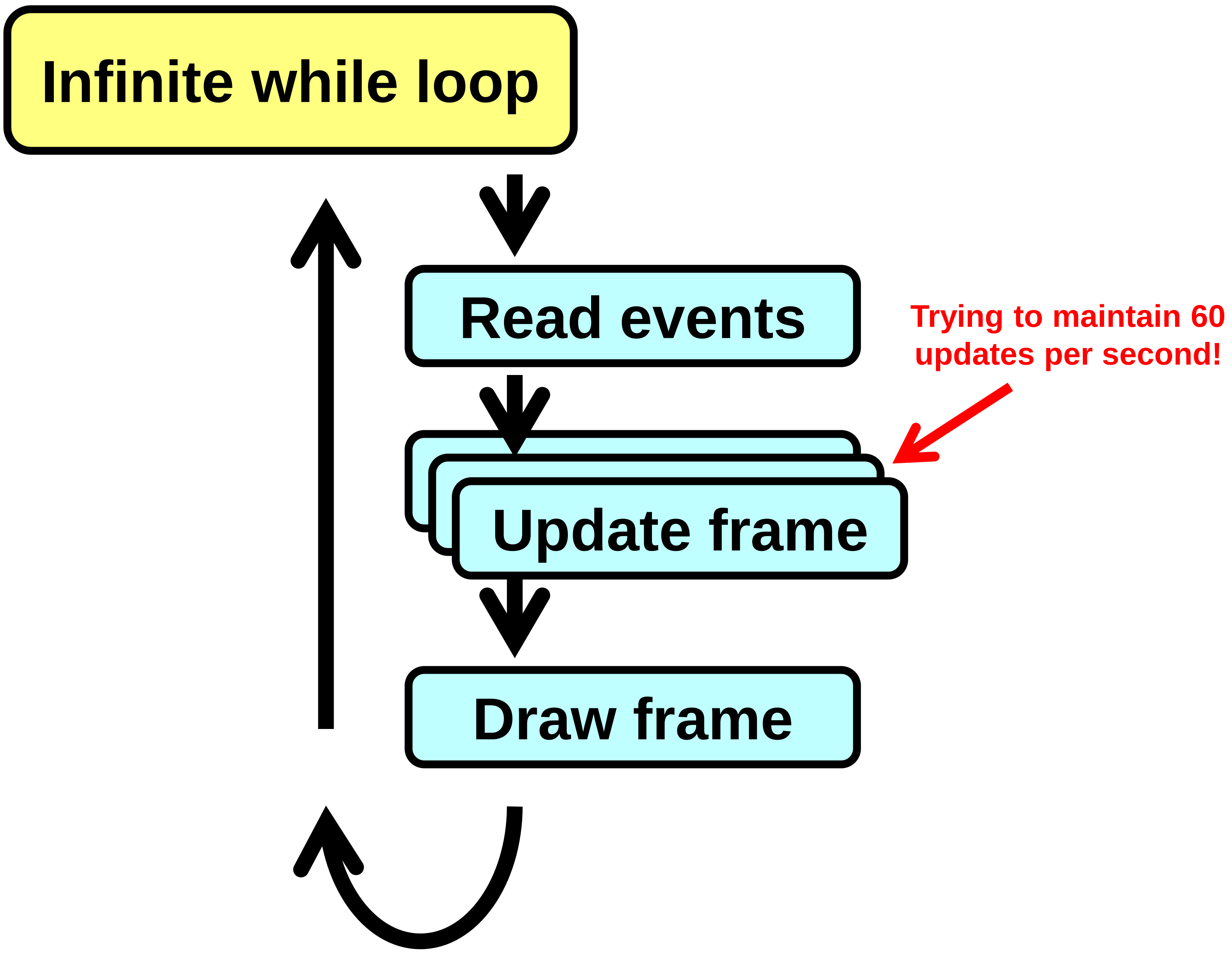
Infinite while loop

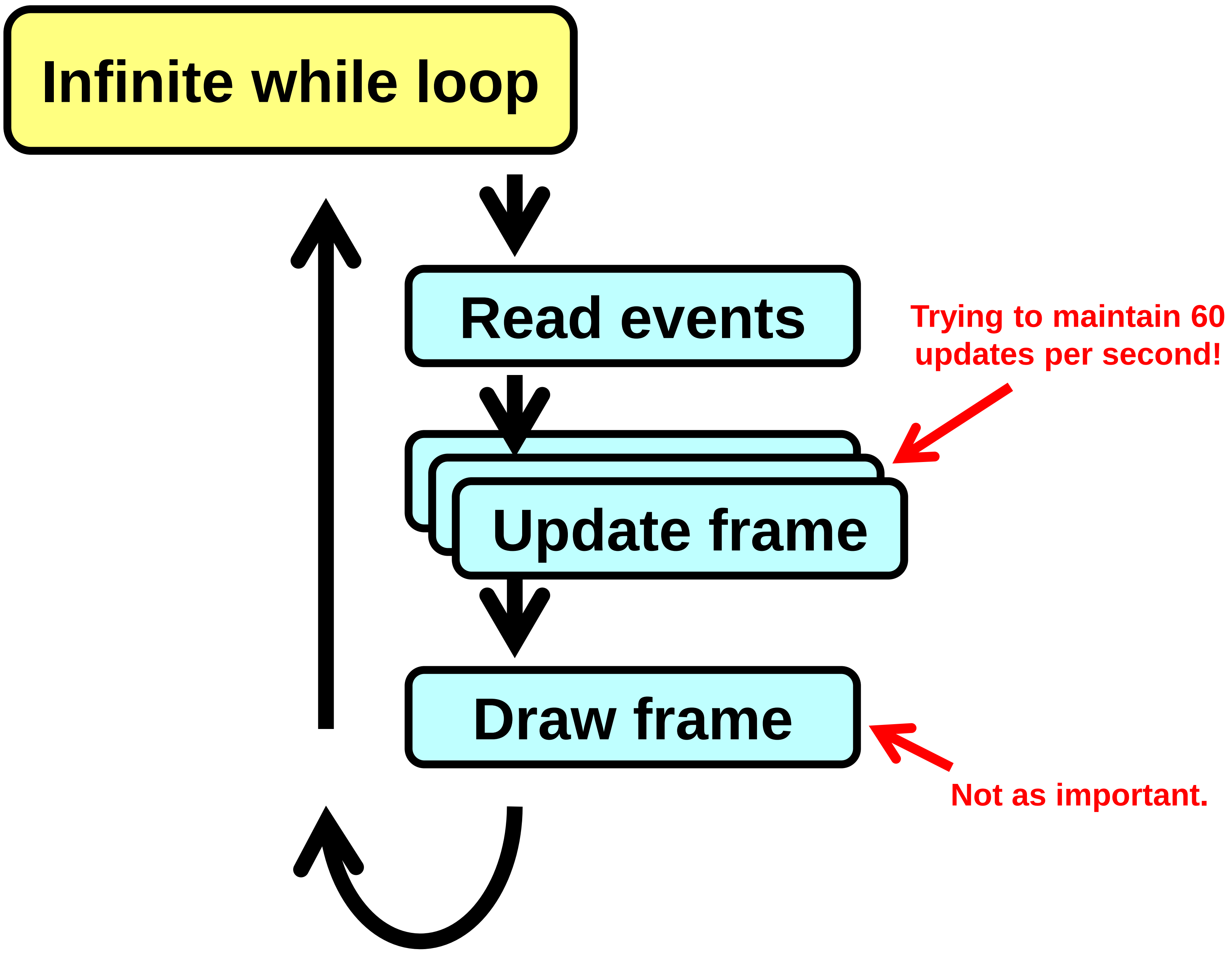



Infinite while loop
Update frame
Draw frame


Infinite while loop
Read events
Update frame
Draw frame

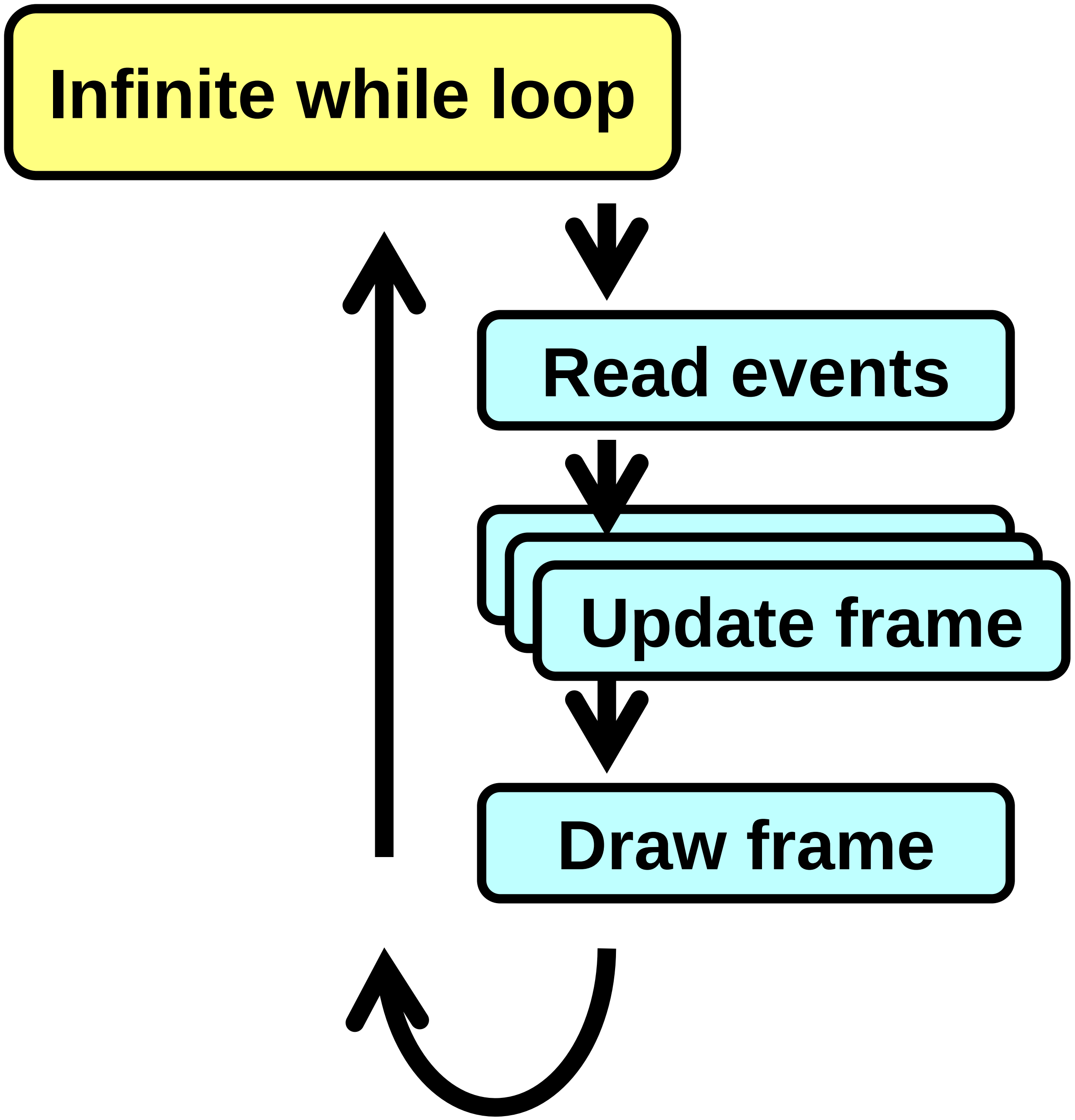

Infinite while loop
Read events
Update frame
Draw frame

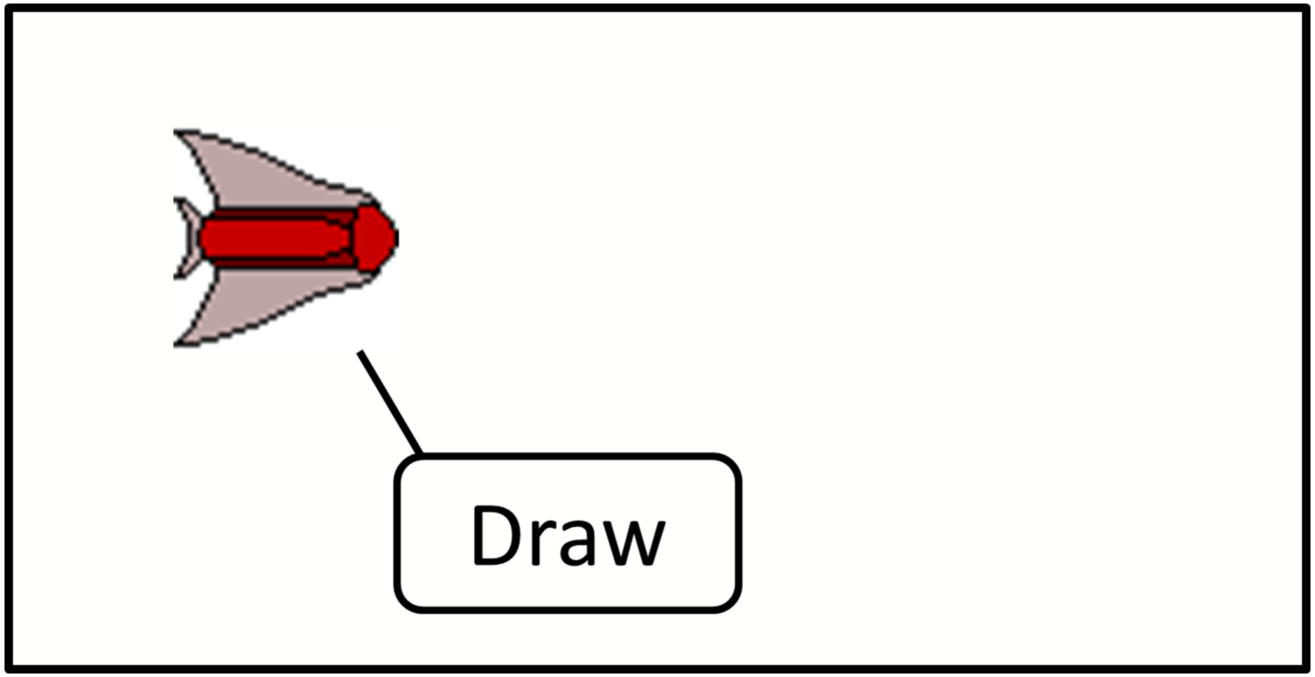

Infinite while loop
Read events
Update frame
Draw frame
Trying to maintain 60 updates per second!

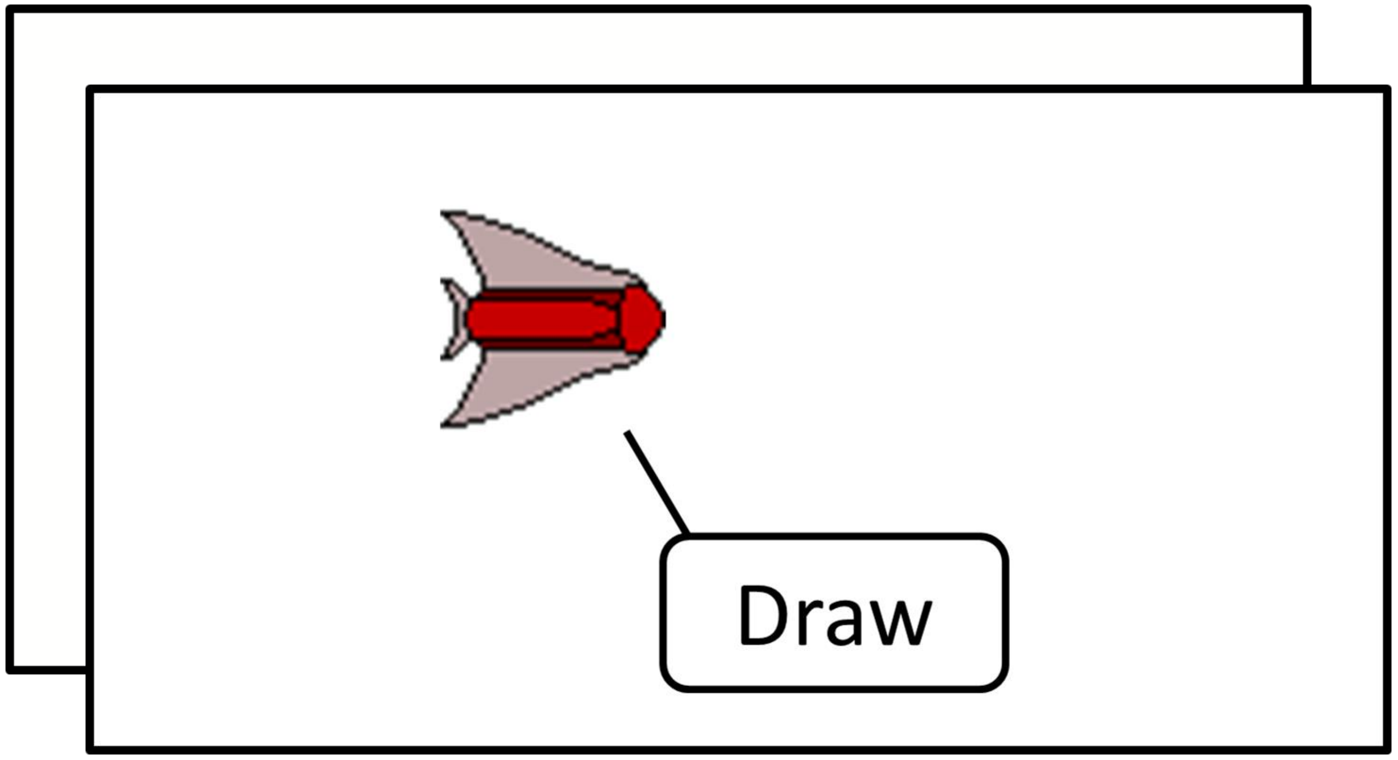

Infinite while loop
Read events
Update frame
Draw frame
Trying to maintain 60 updates per second!
Not as important.

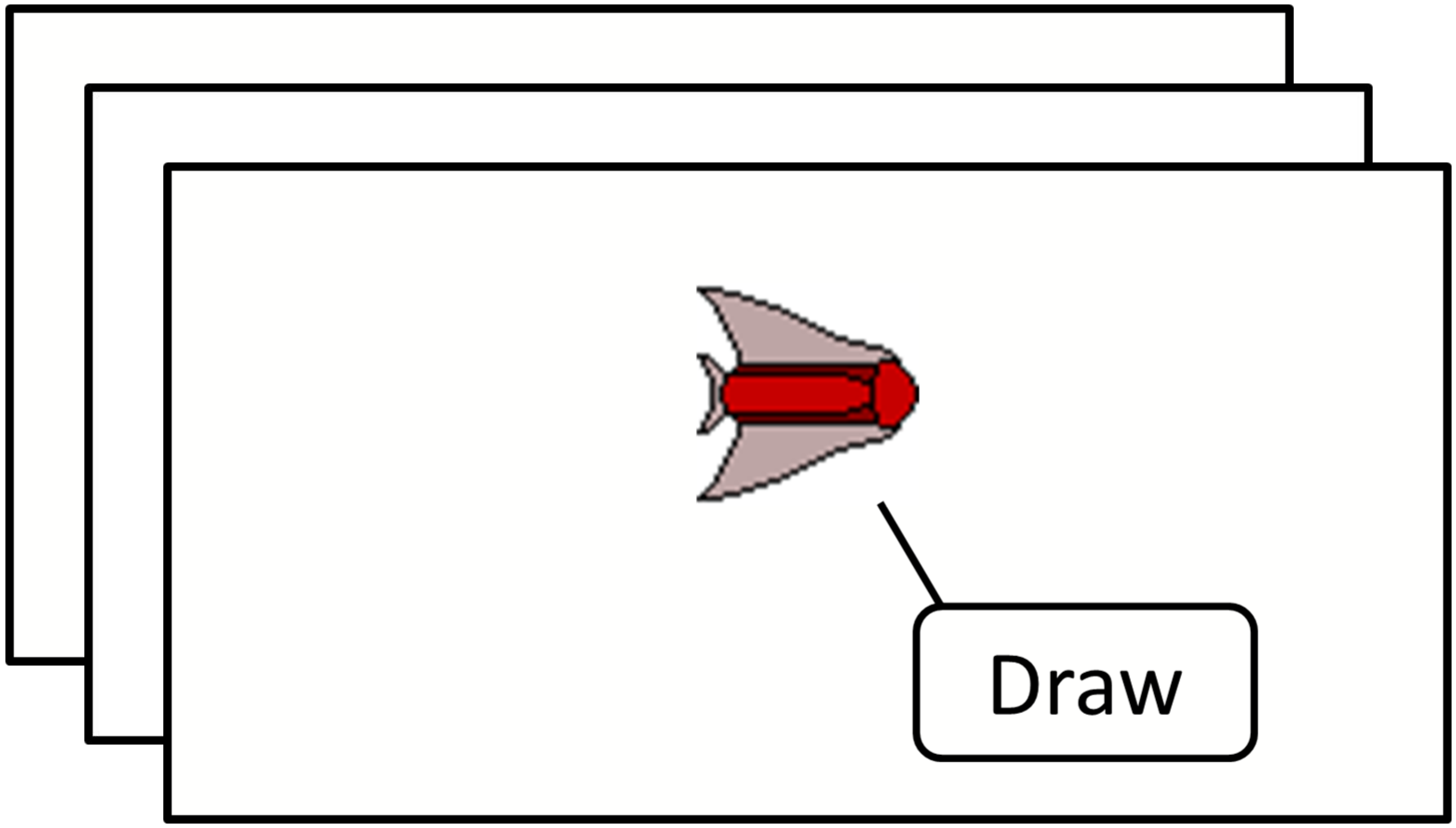
V1

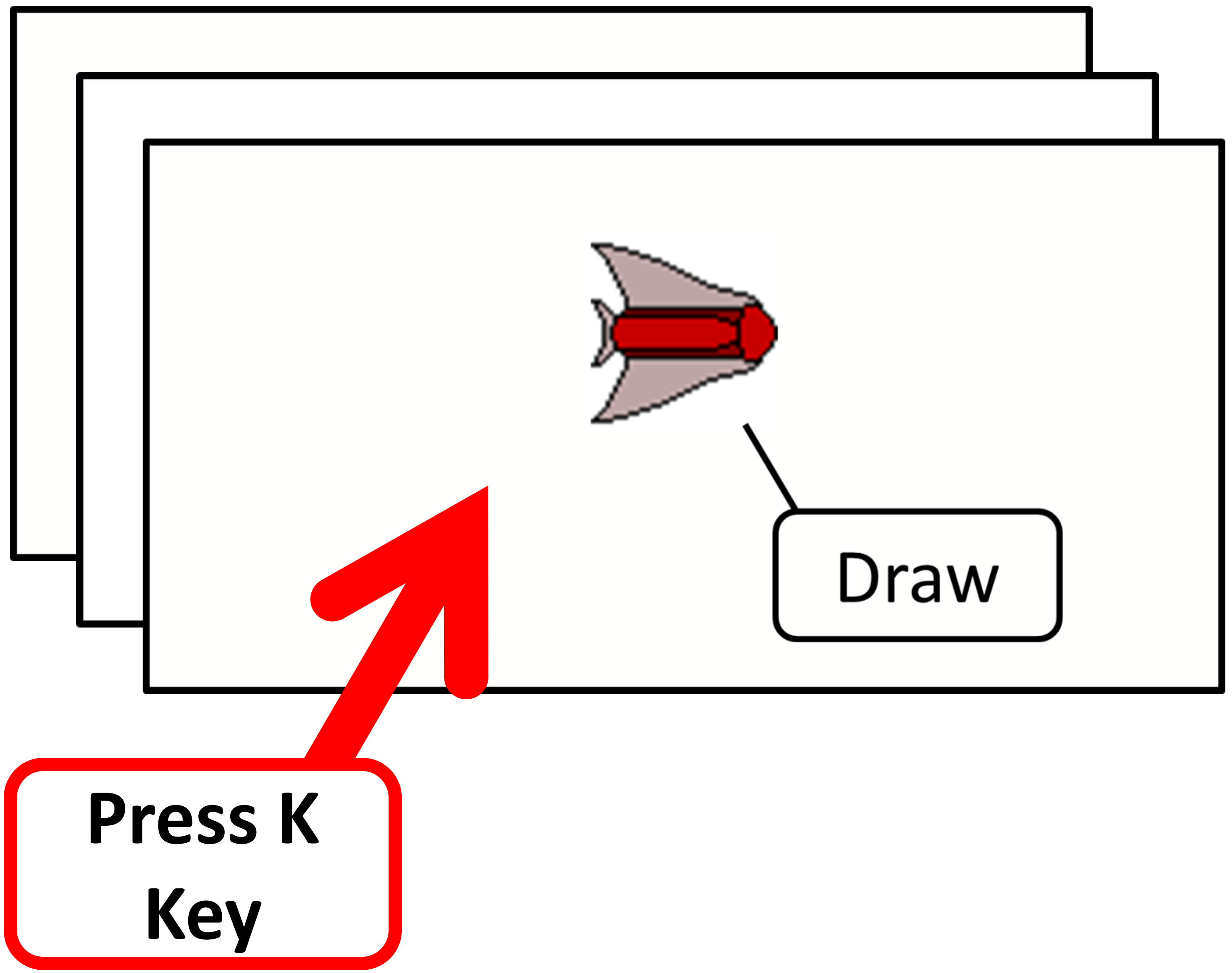
Update and Draw frames

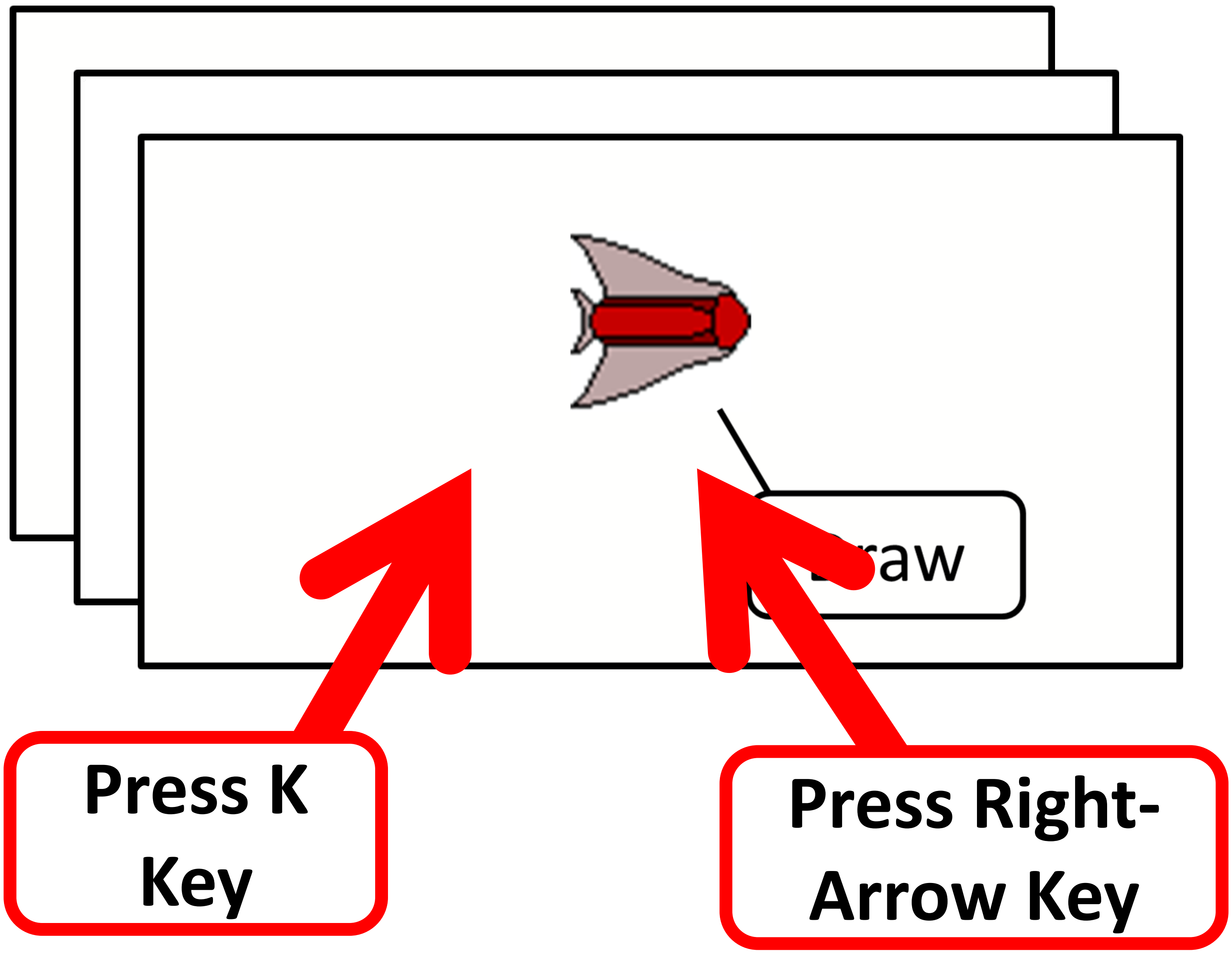

Infinite while loop
Read events
Update frame
Draw frame

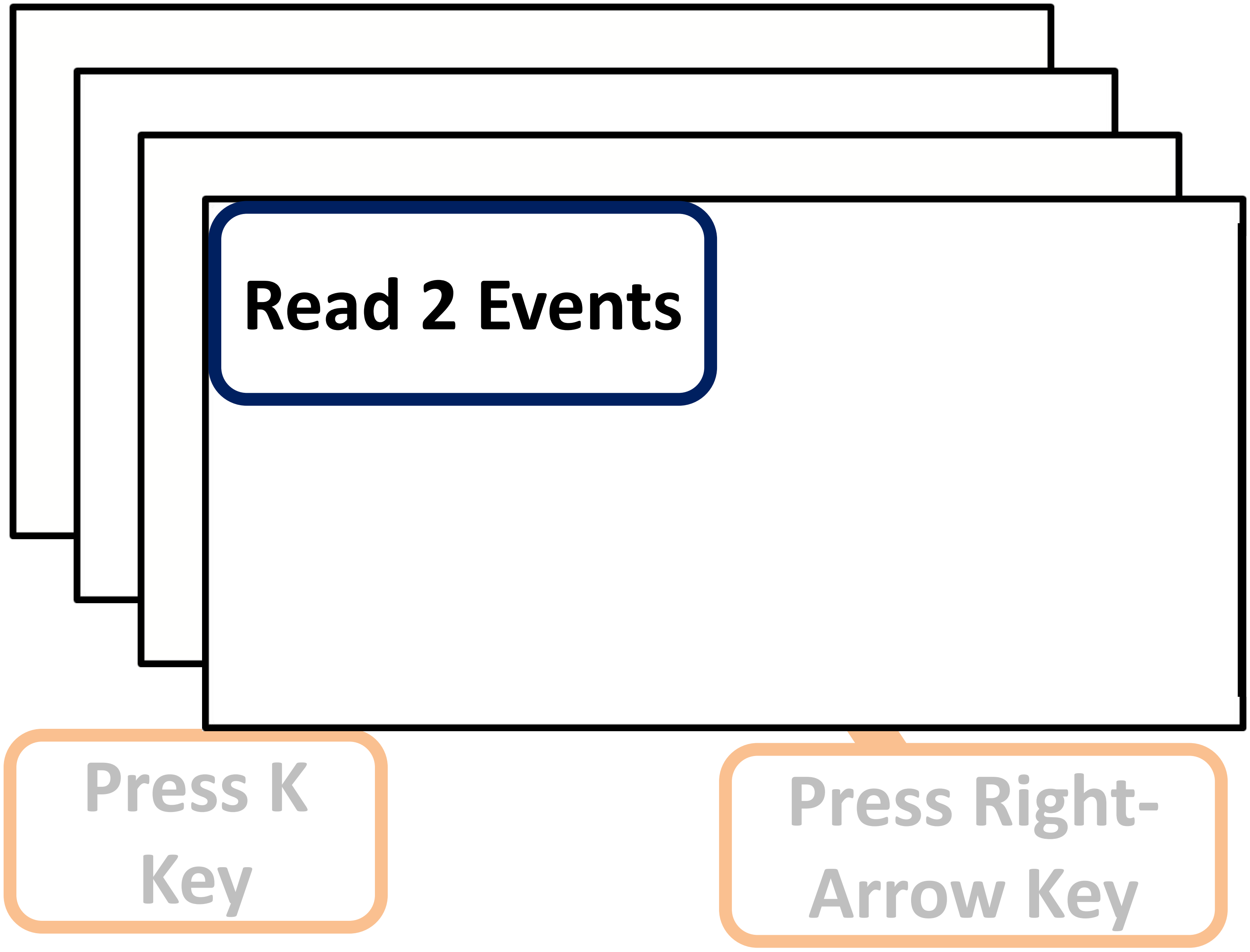

Draw

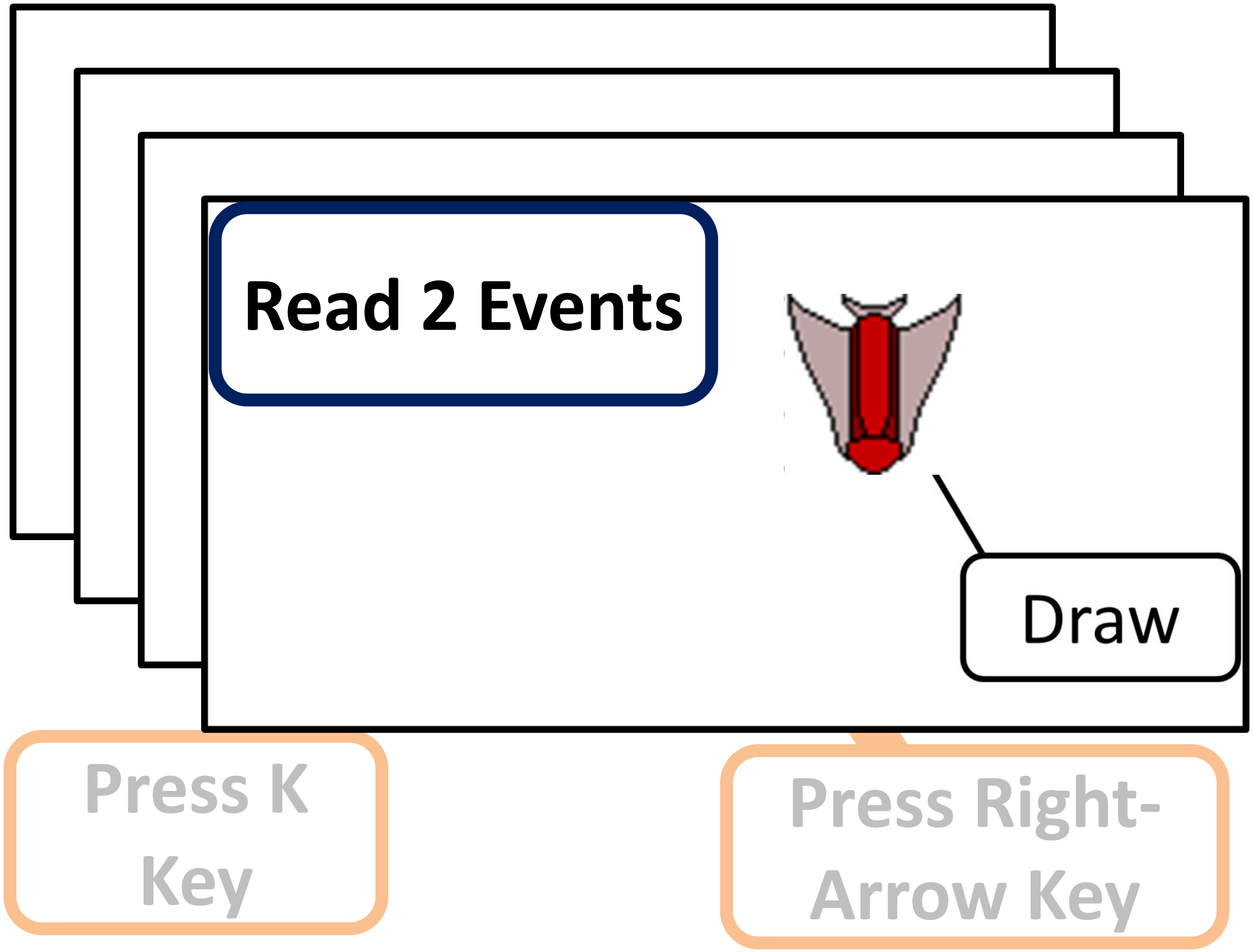

Draw

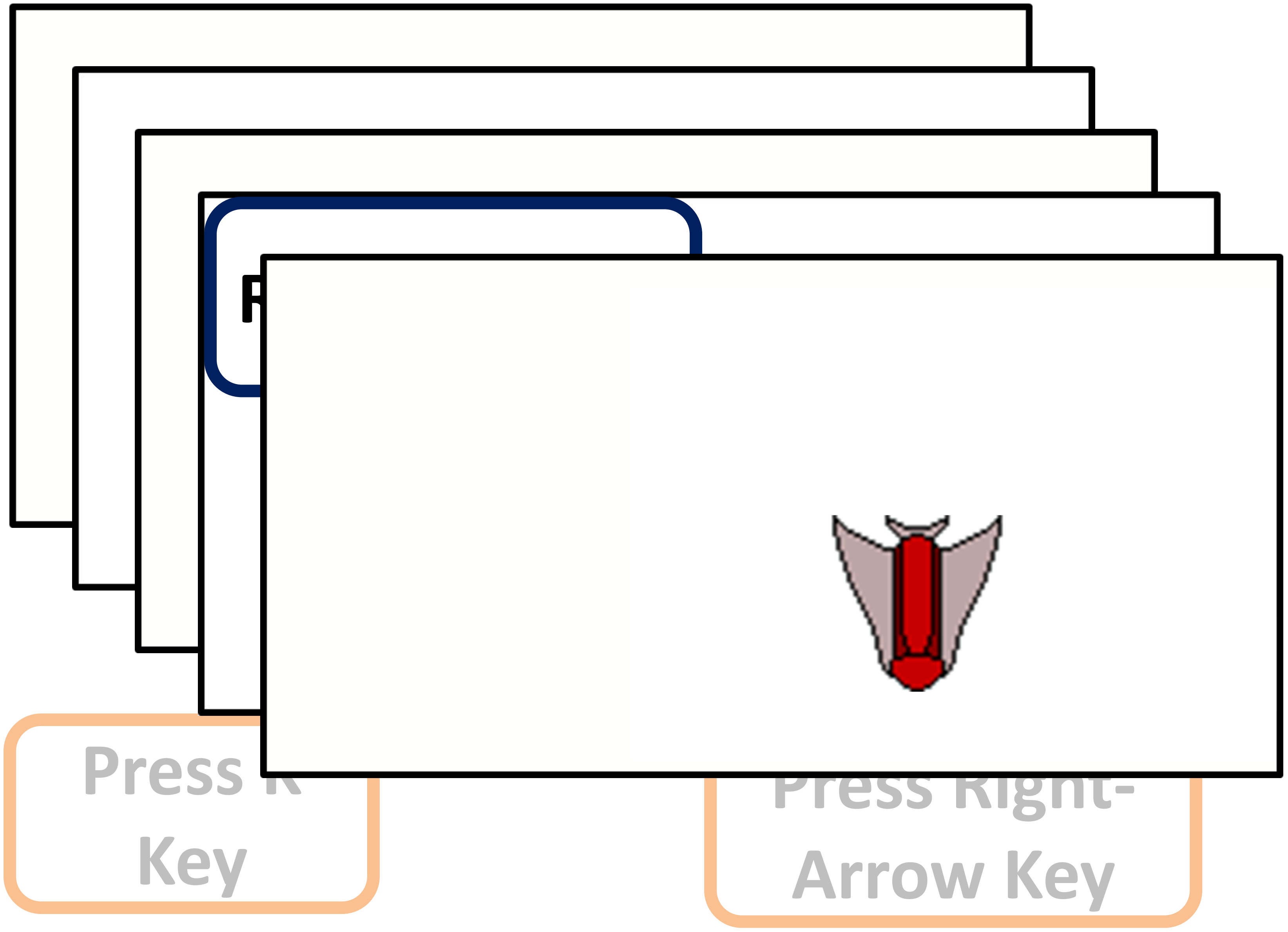

Draw


Draw
Press K Key


Draw
Press K Key
Press Right-Arrow Key

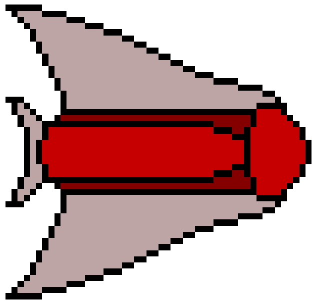

Read 2 Events
Press K Key
Press Right-Arrow Key


Read 2 Events
Draw
Press K Key
Press Right-Arrow Key


Press R
Key
Press Right-
Arrow Key

V2

Basic Keyboard Input

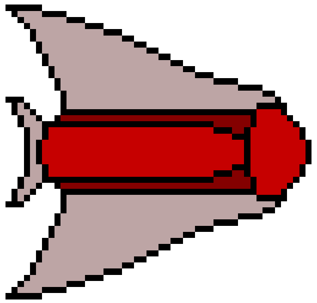


int x

int y

int vx

int vy

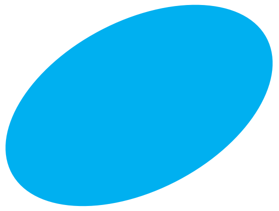


int x

int y

int vx

int vy

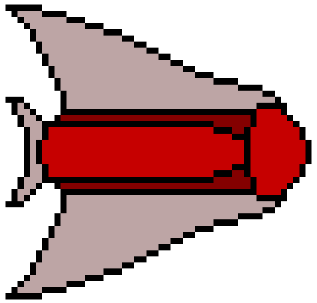


int x

int y

int vx

int vy

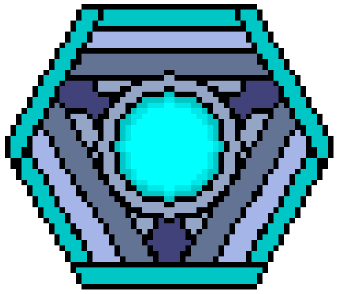


int x

int y

int vx

int vy

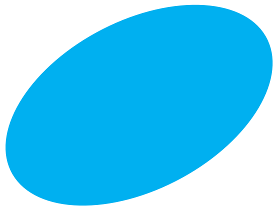


int x

int y

int vx

int vy



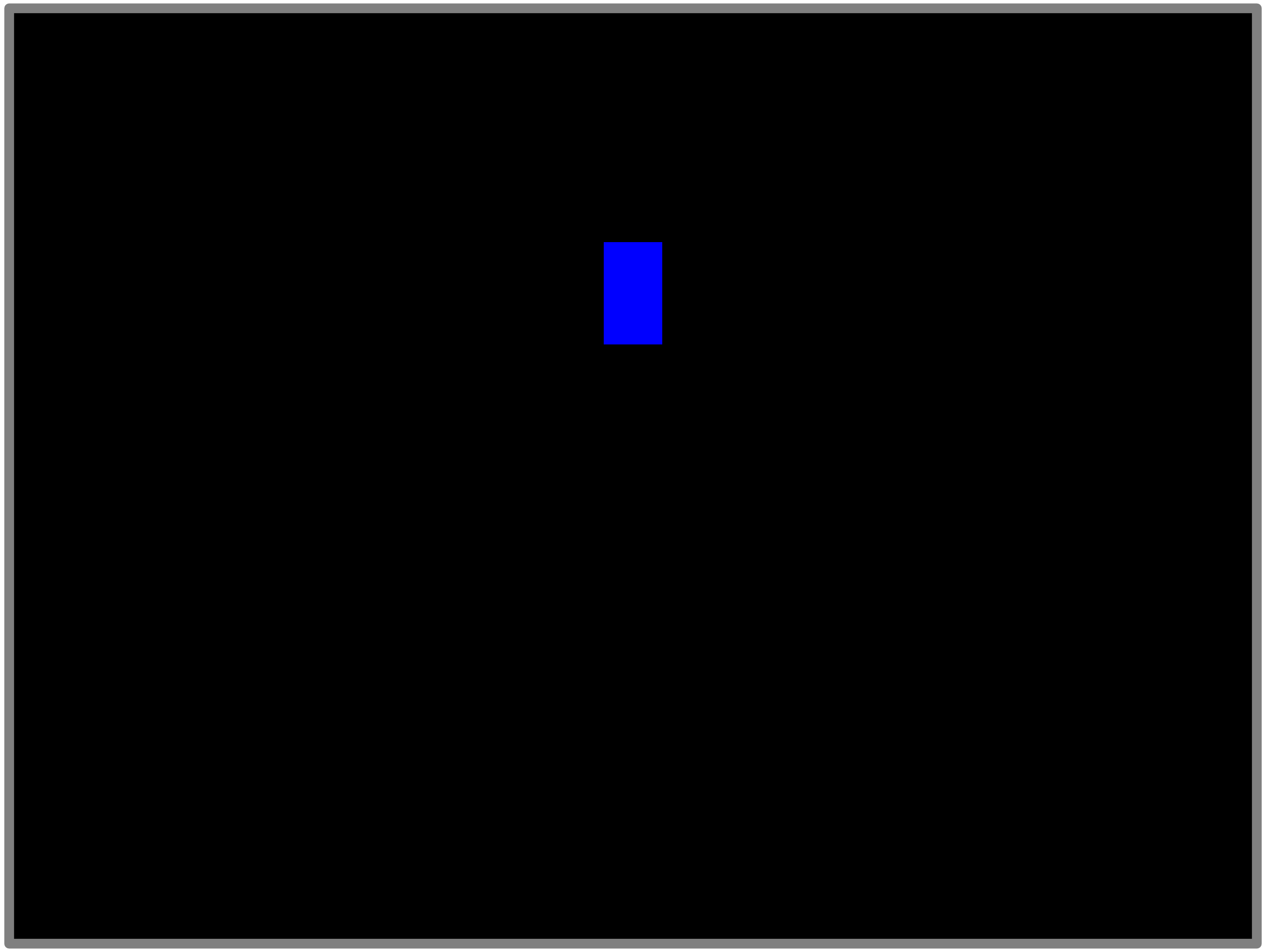
int x

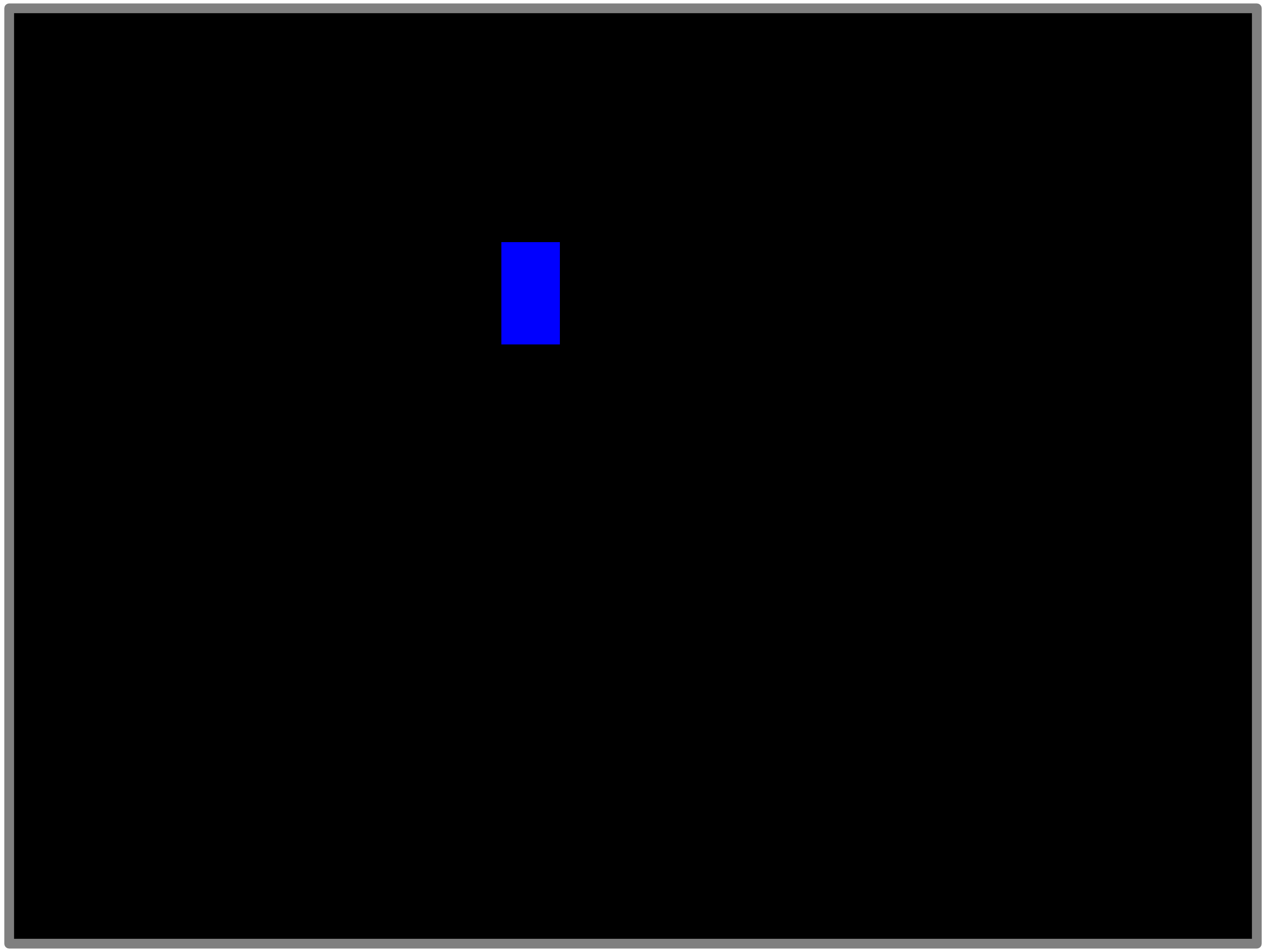
int y

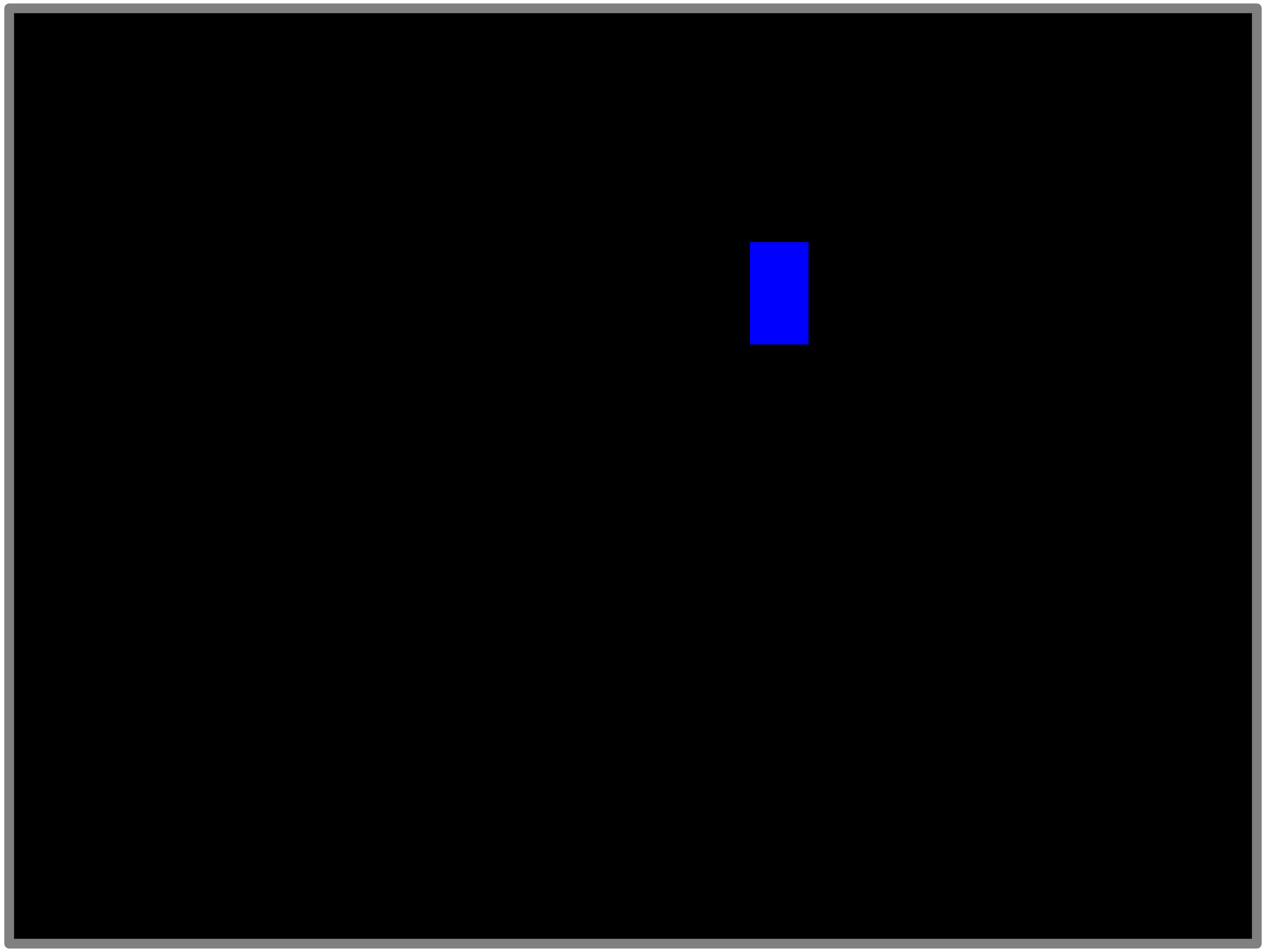
int vx

int vy

Player



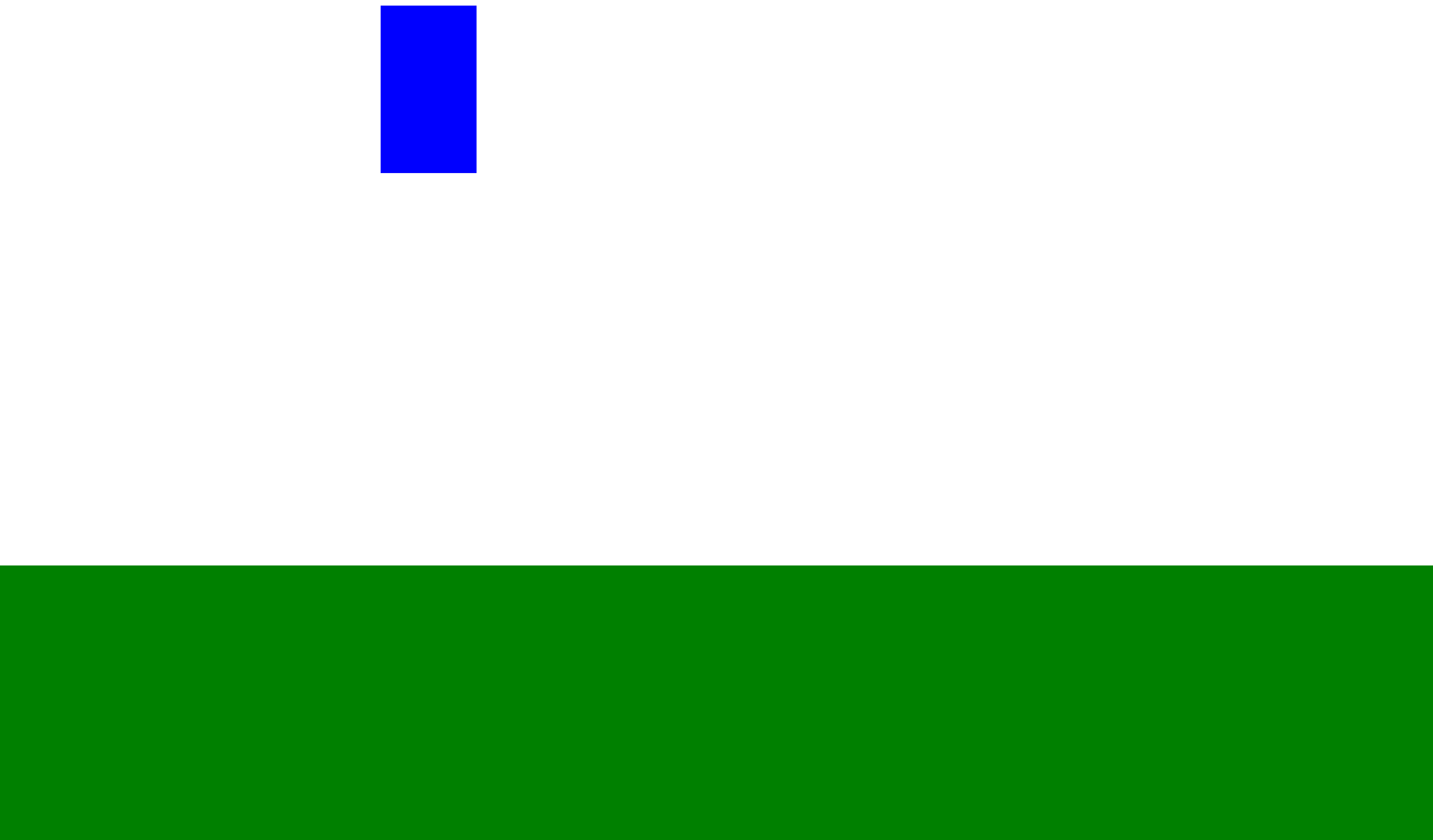


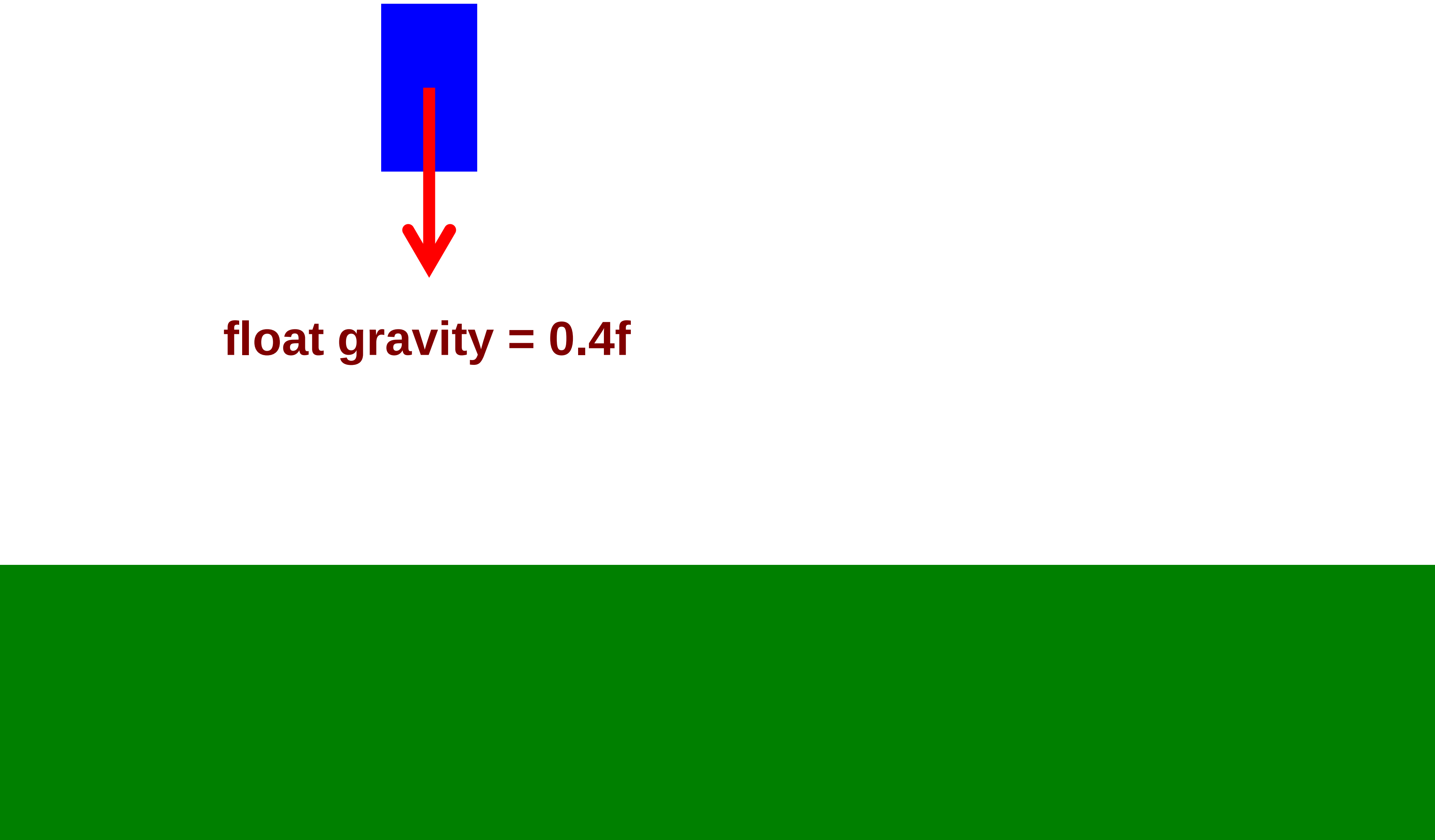


V3

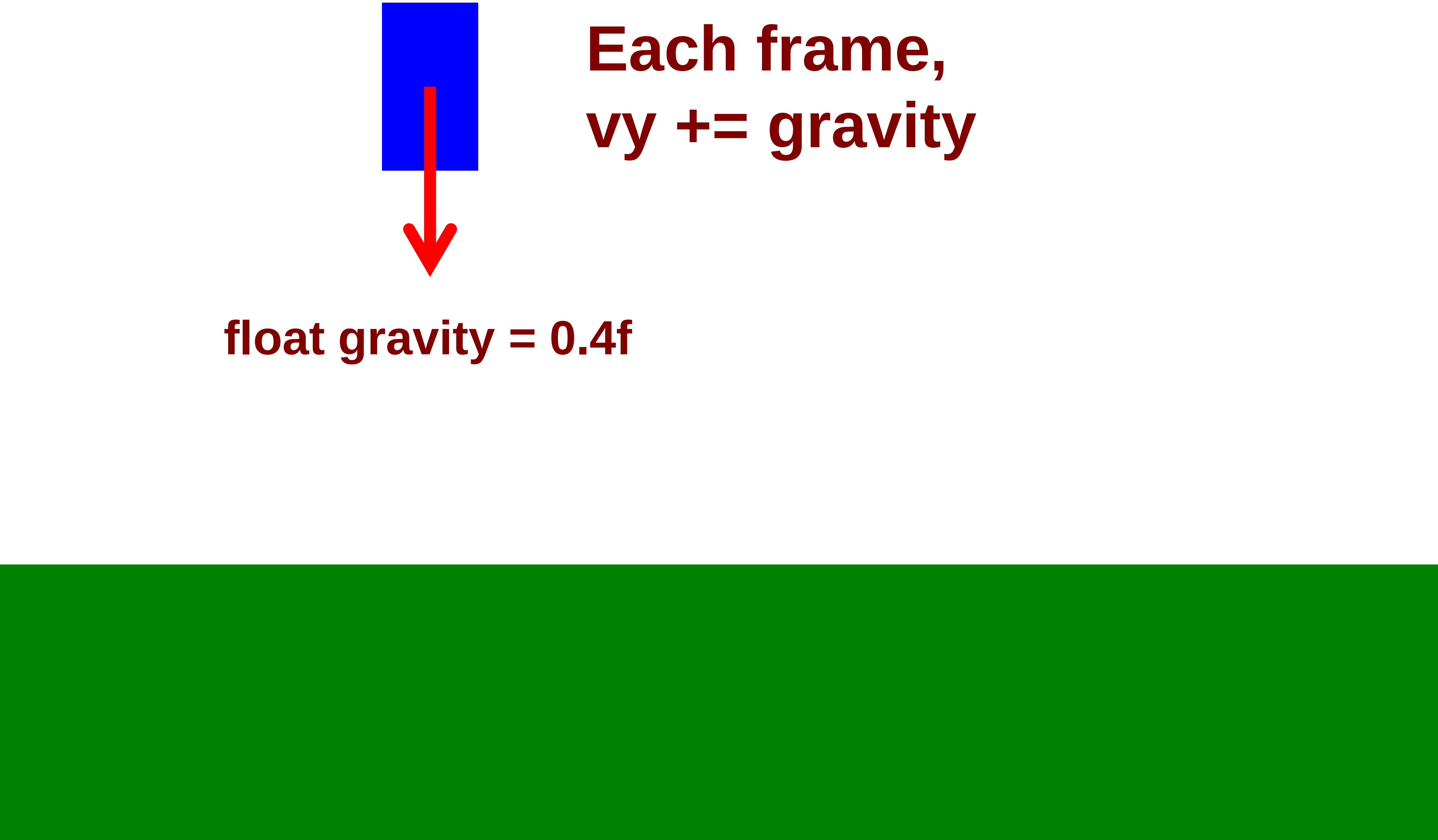
Player Class

Gravity



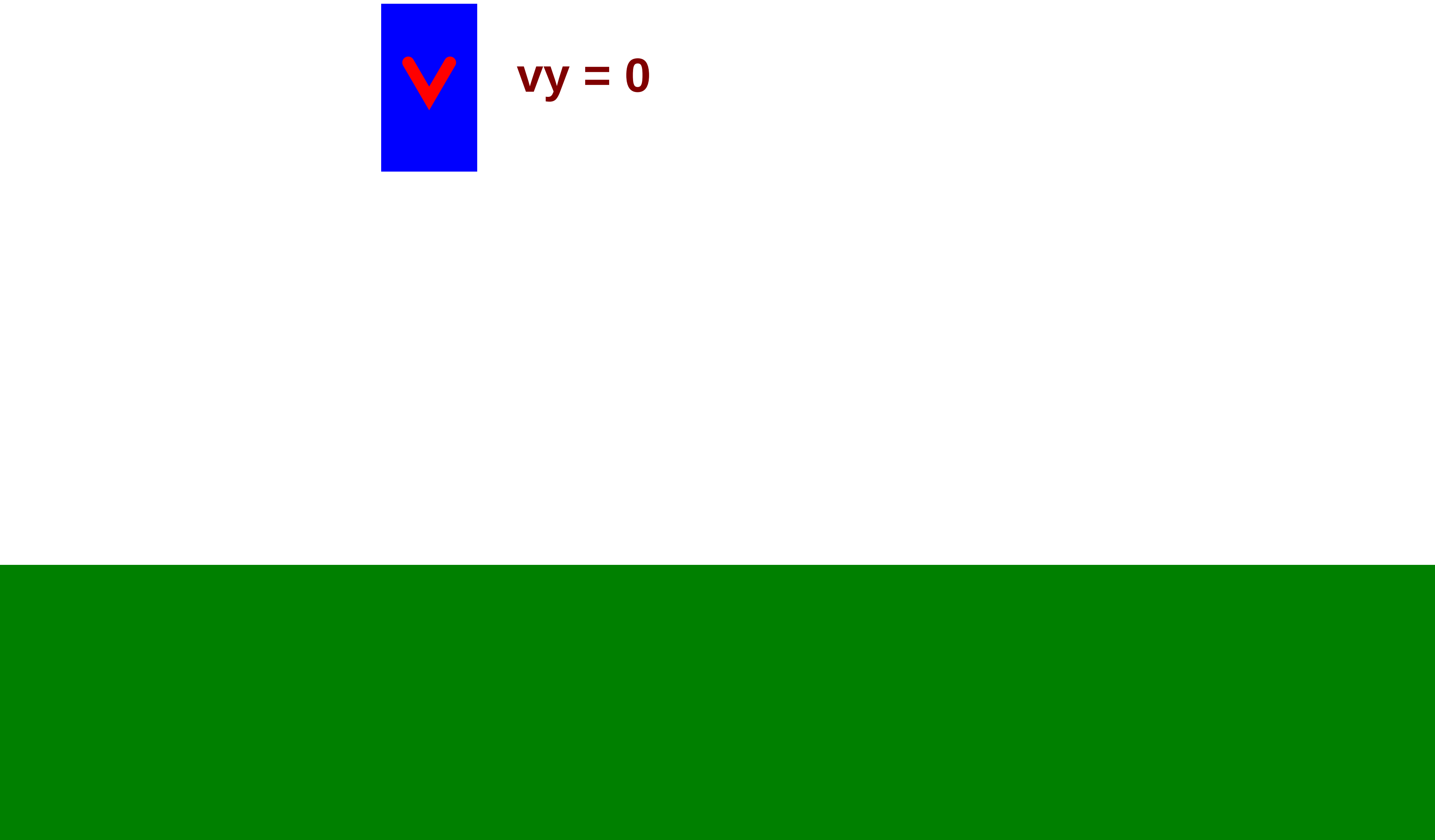


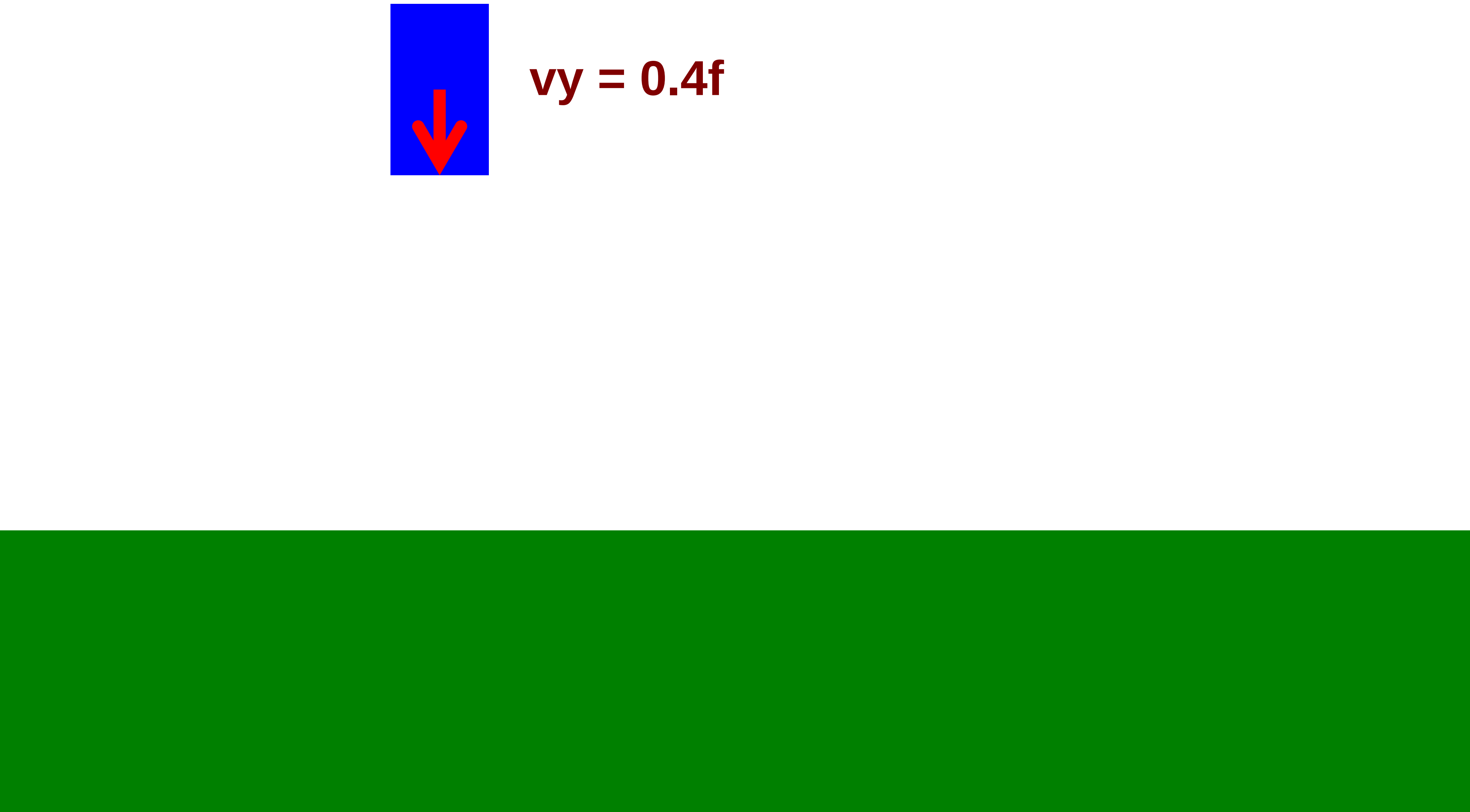
```
float gravity = 0.4f
```

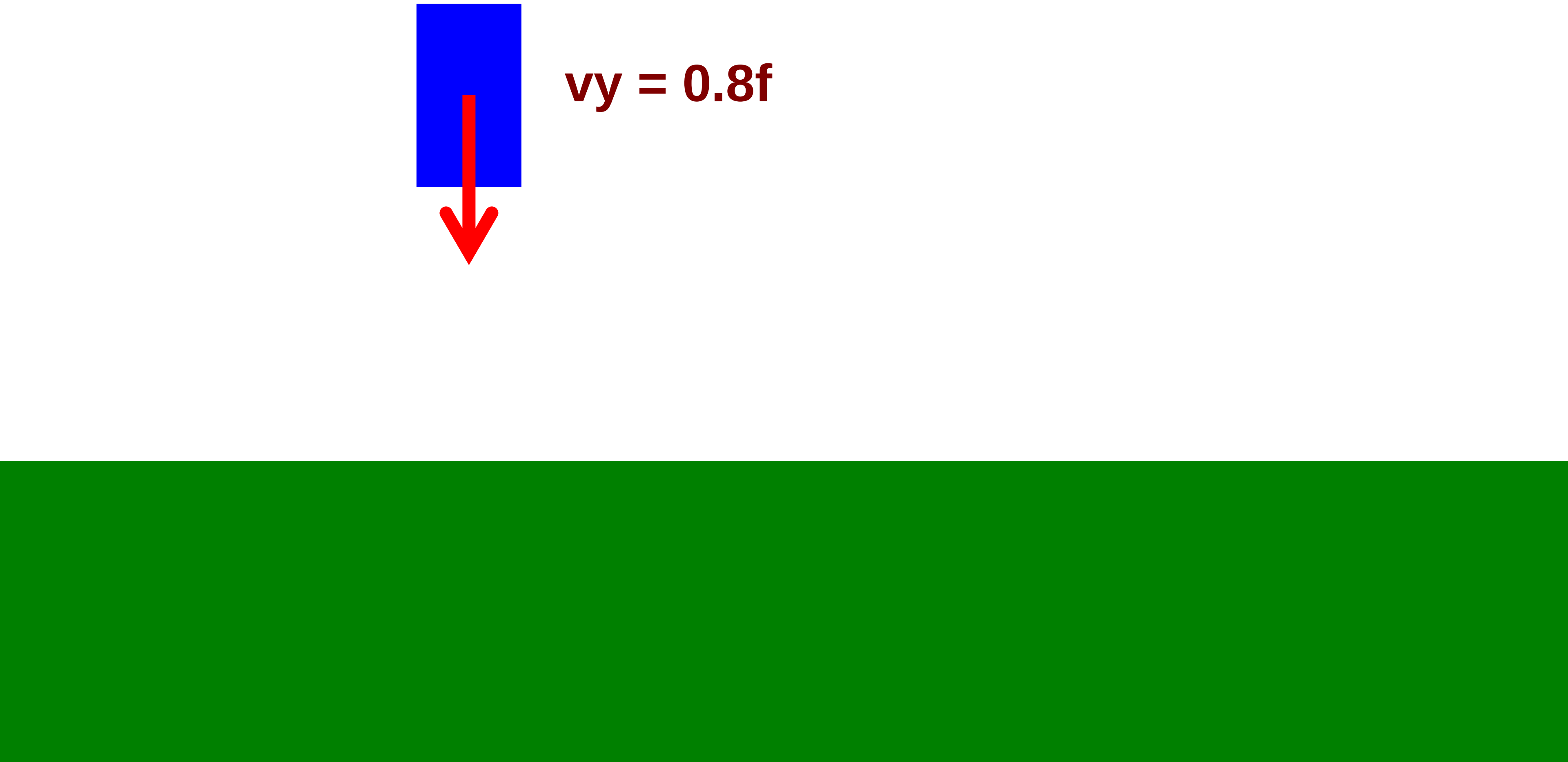


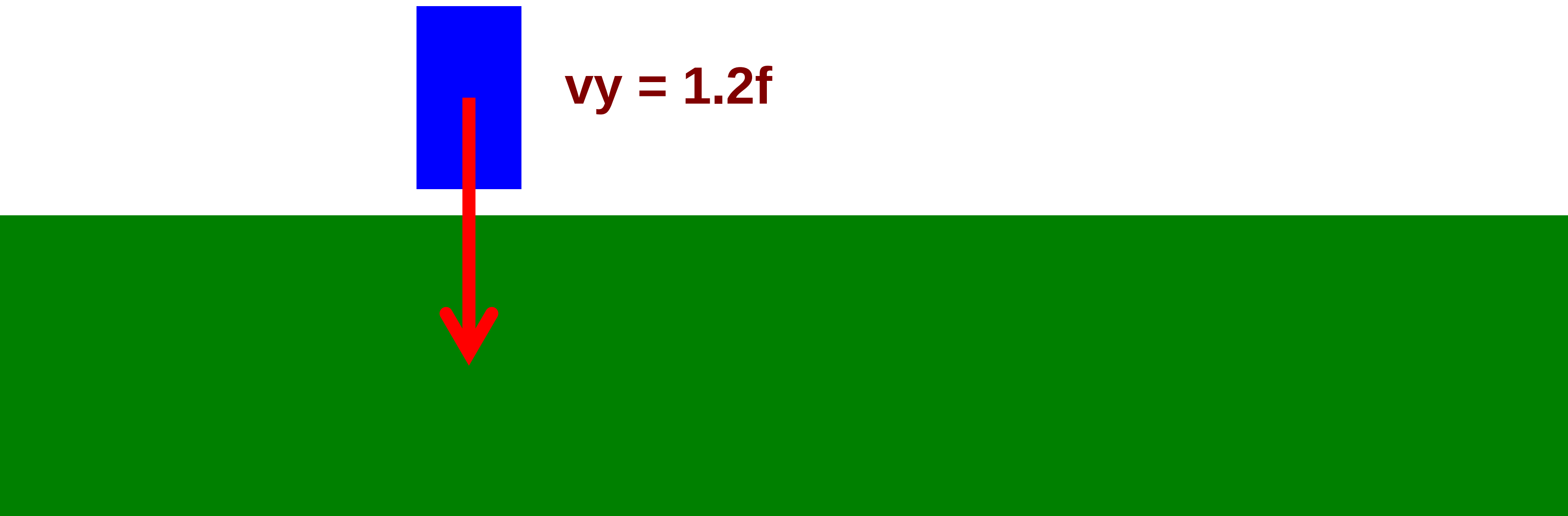
Each frame,
vy += gravity

float gravity = 0.4f

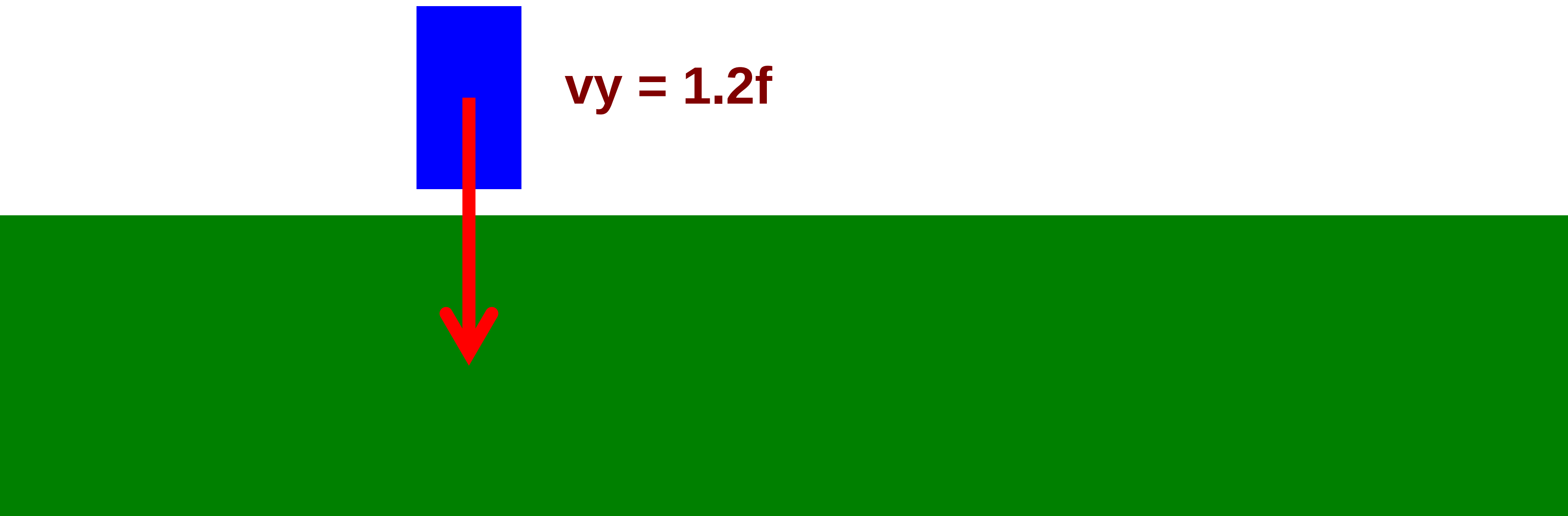

vy = 0

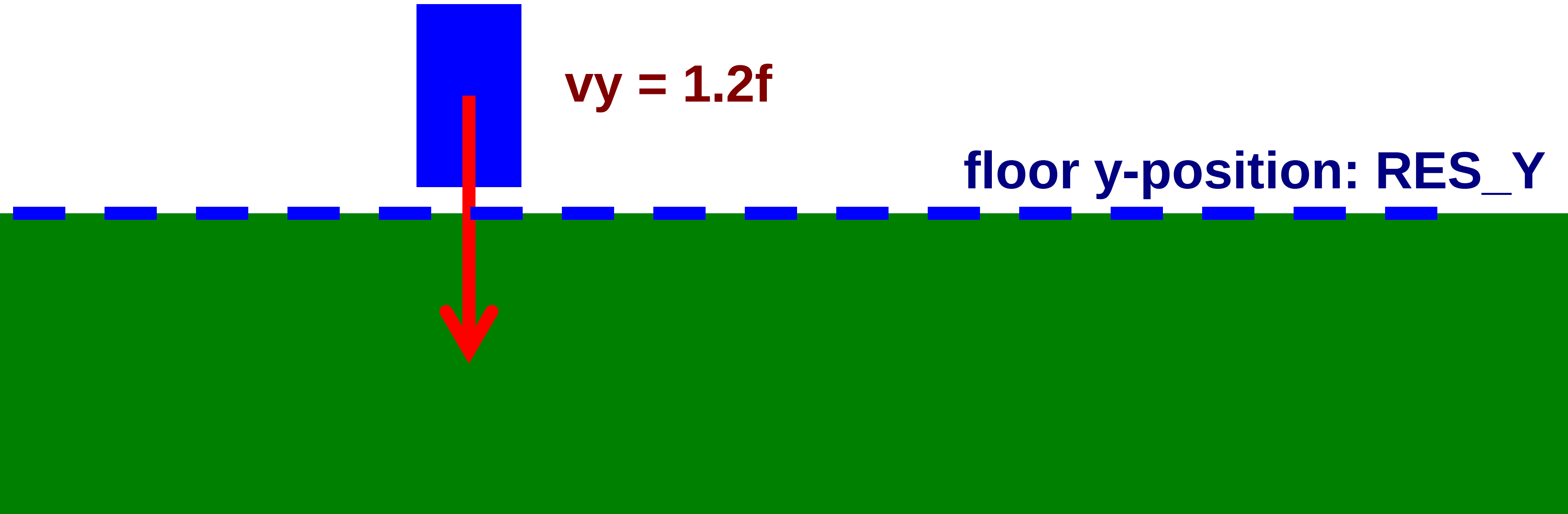

vy = 0.4f

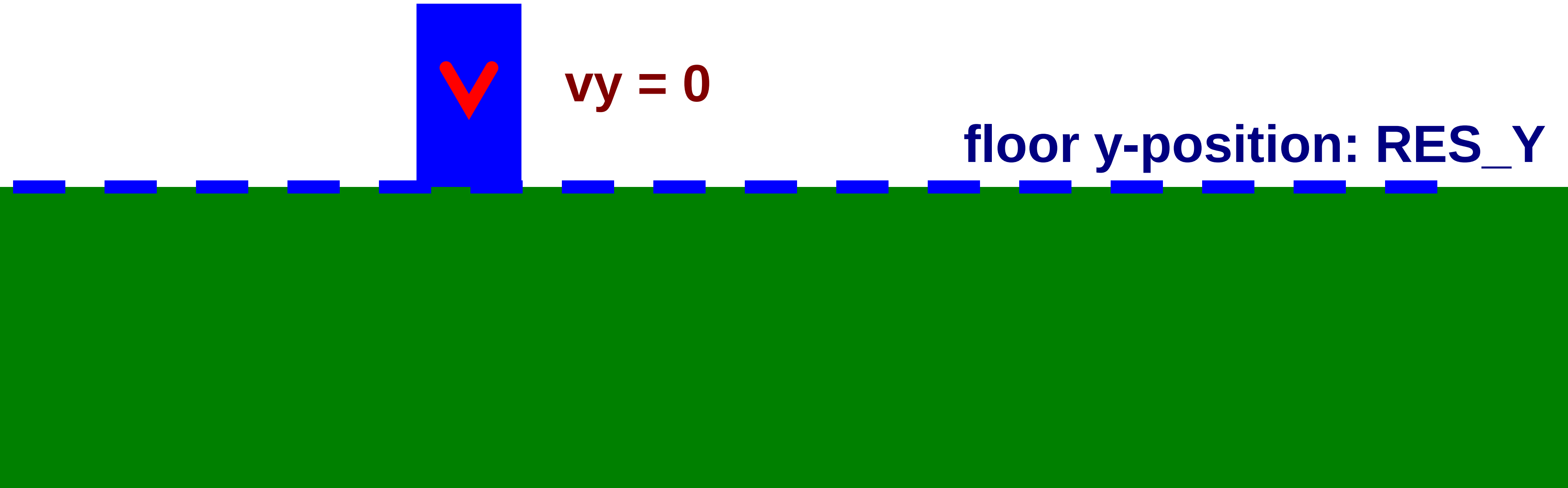

vy = 0.8f

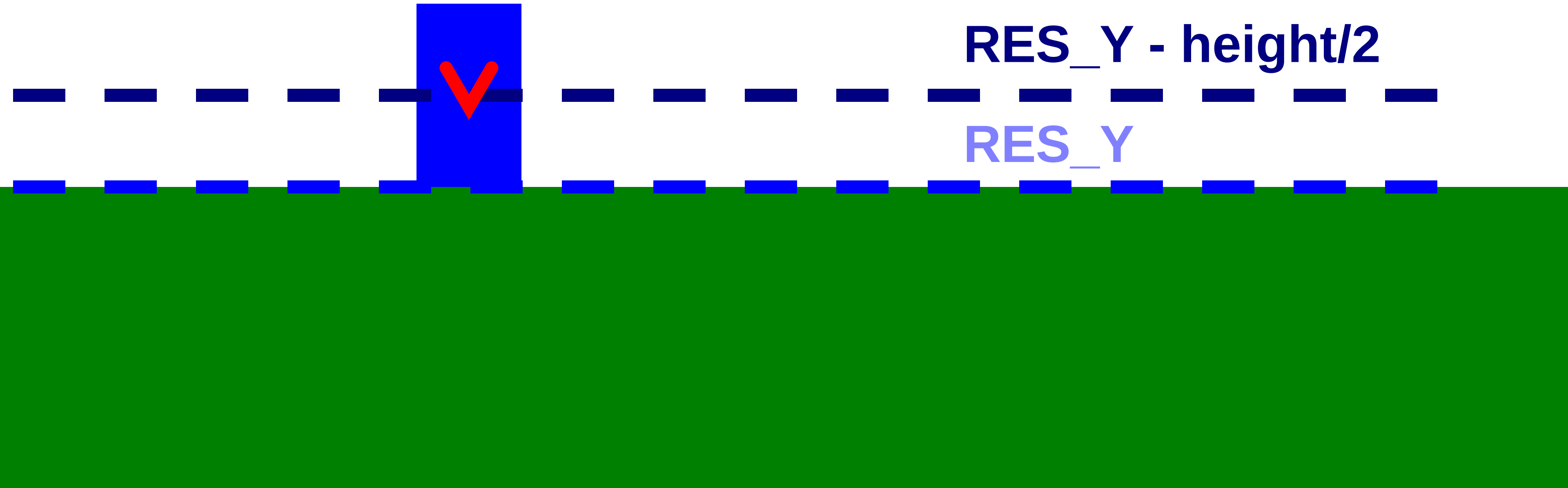

vy = 1.2f

vy = 1.6f

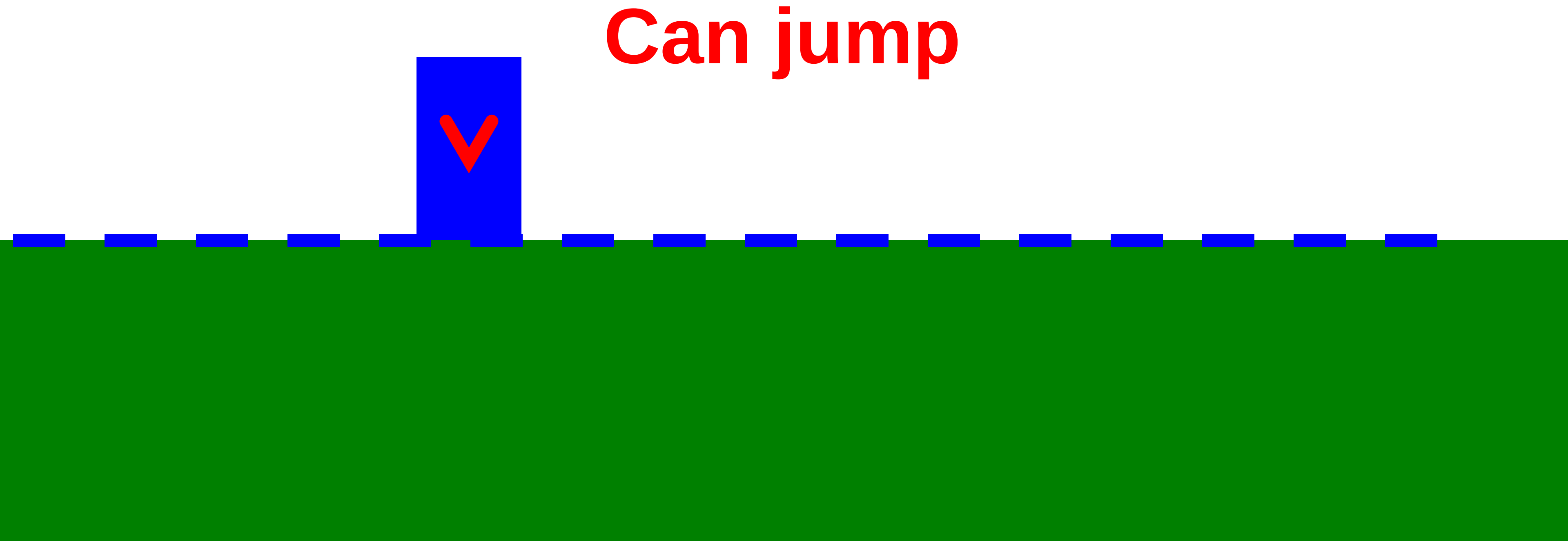

vy = 1.2f

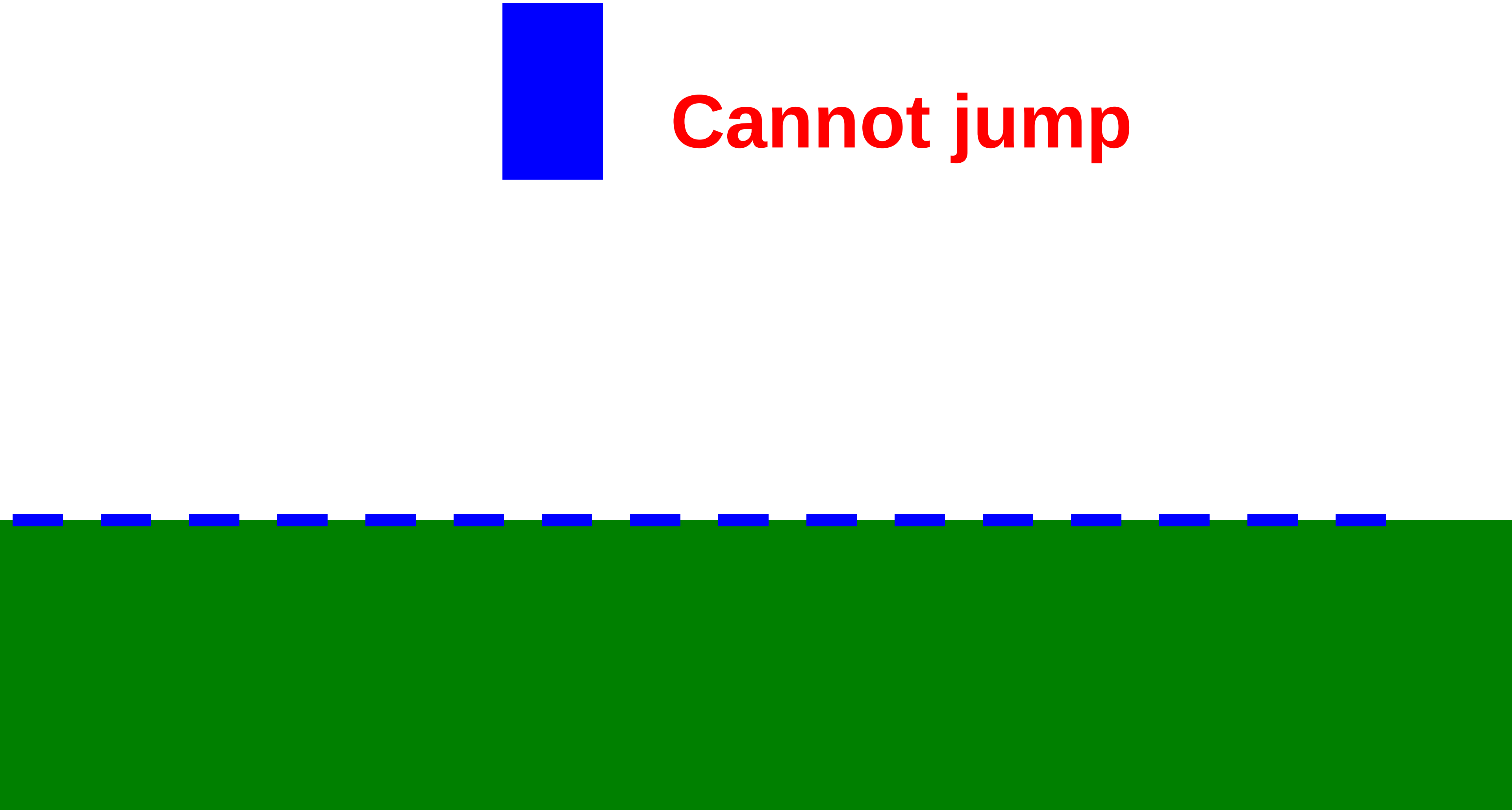

vy = 1.2f
floor y-position: RES_Y


vy = 0
floor y-position: RES_Y

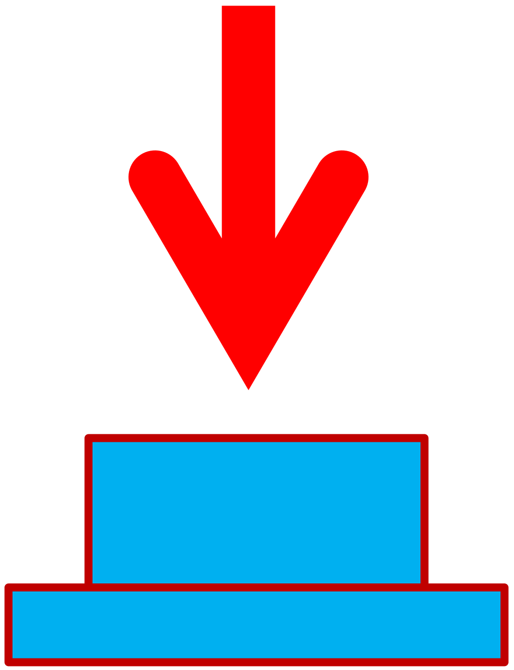

RES_Y - height/2
RES_Y

Jumping

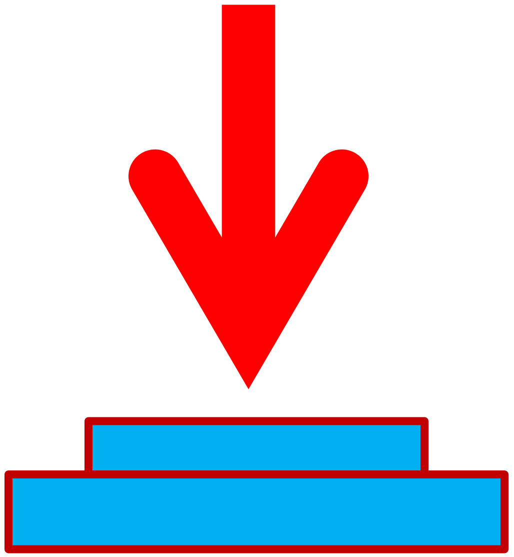

Can jump


Cannot jump

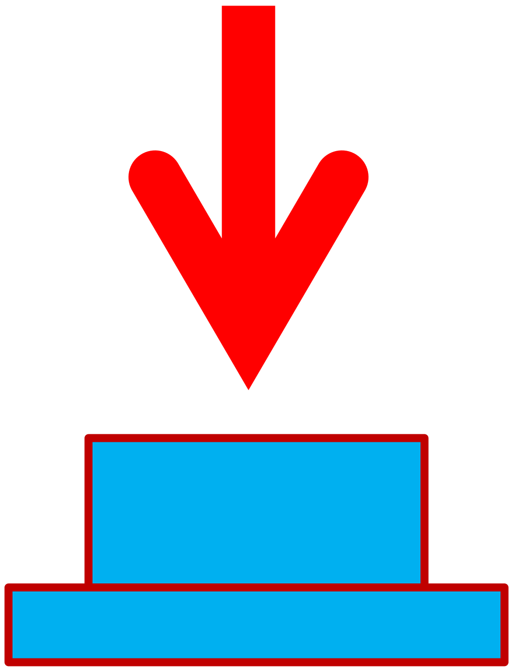
Keyboard:
Track “key being held”



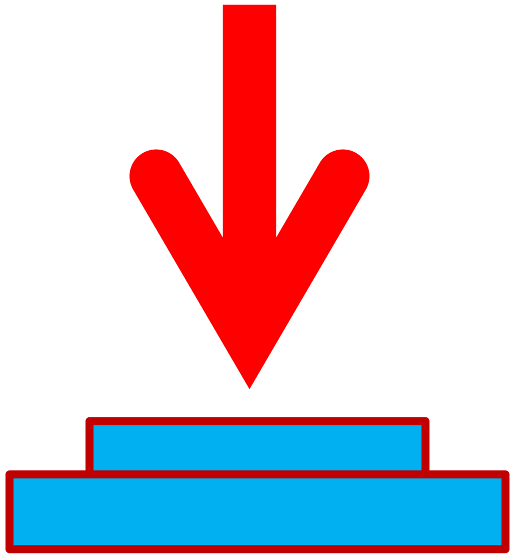
key = False



key = True

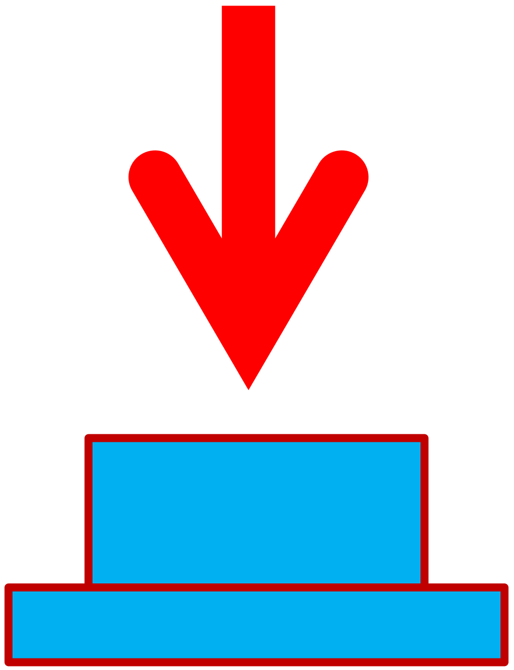


key = False

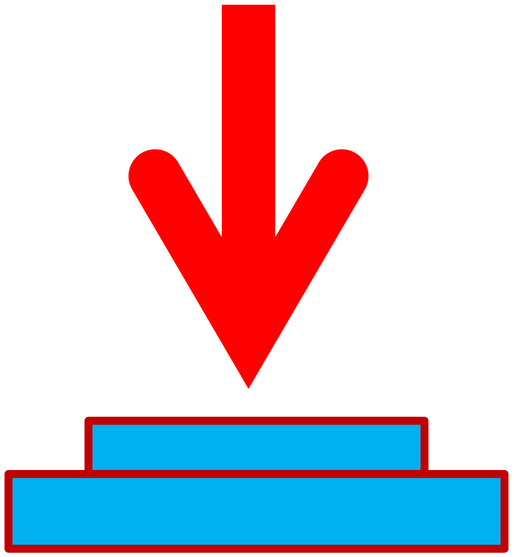


key = True

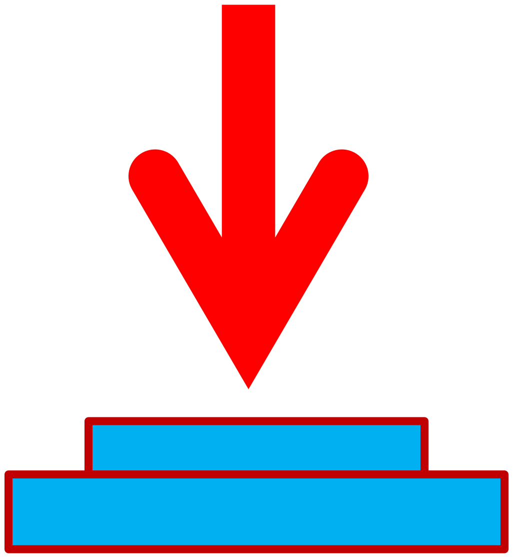
Keyboard:
Track “key click once”



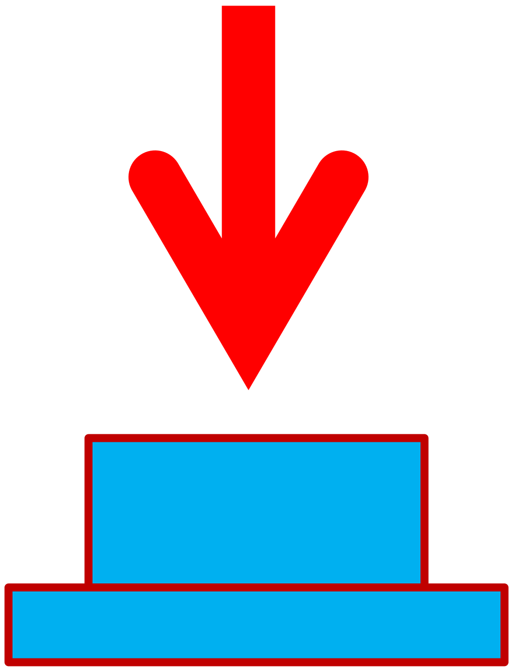
key = False



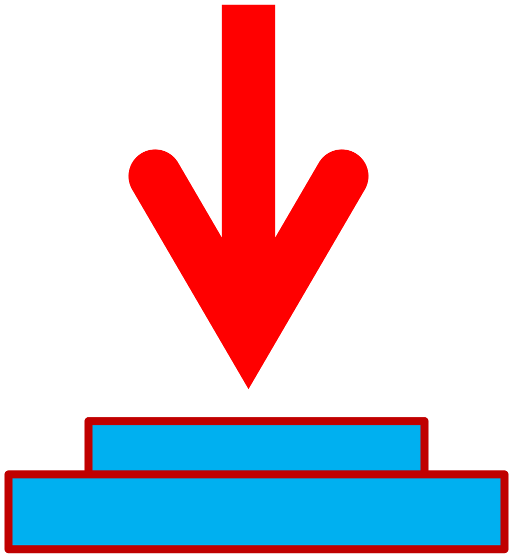
key = True



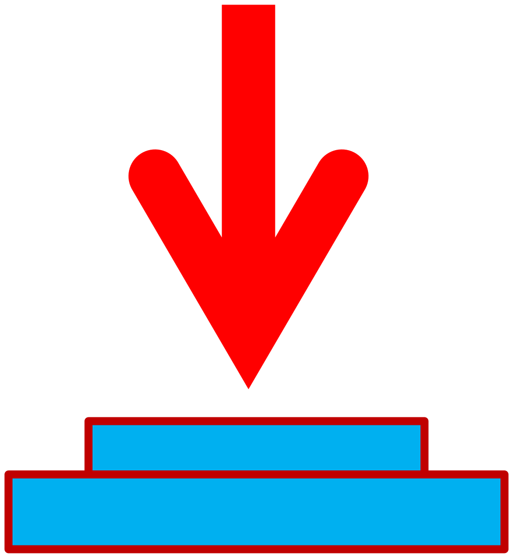
key = False



key = False



key = True

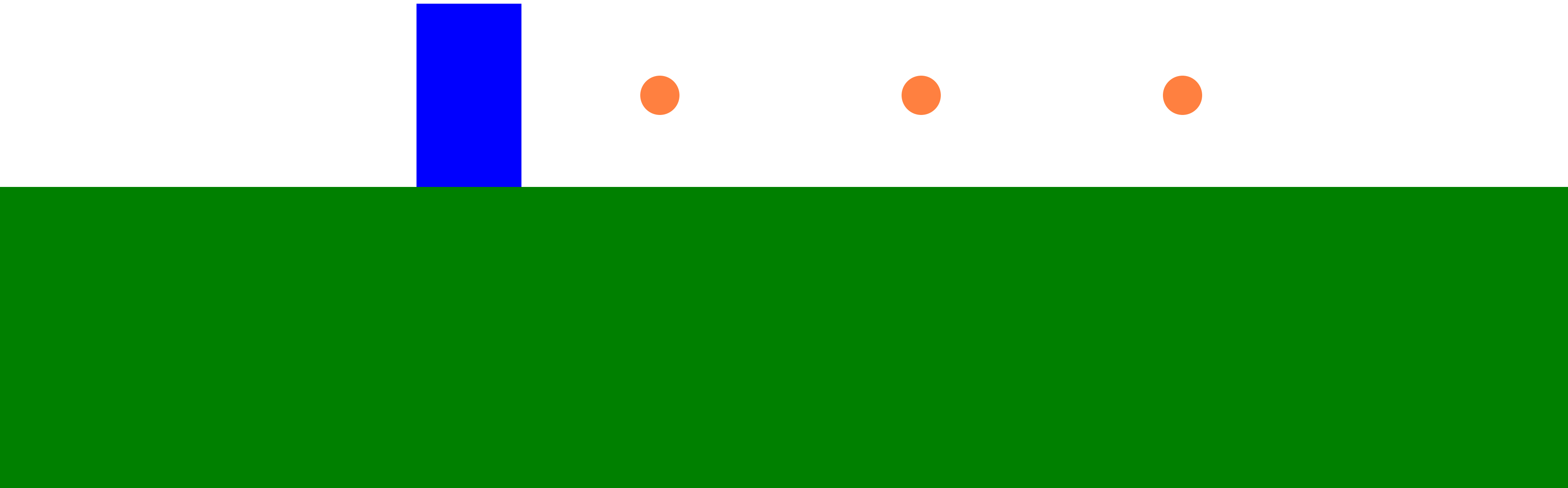


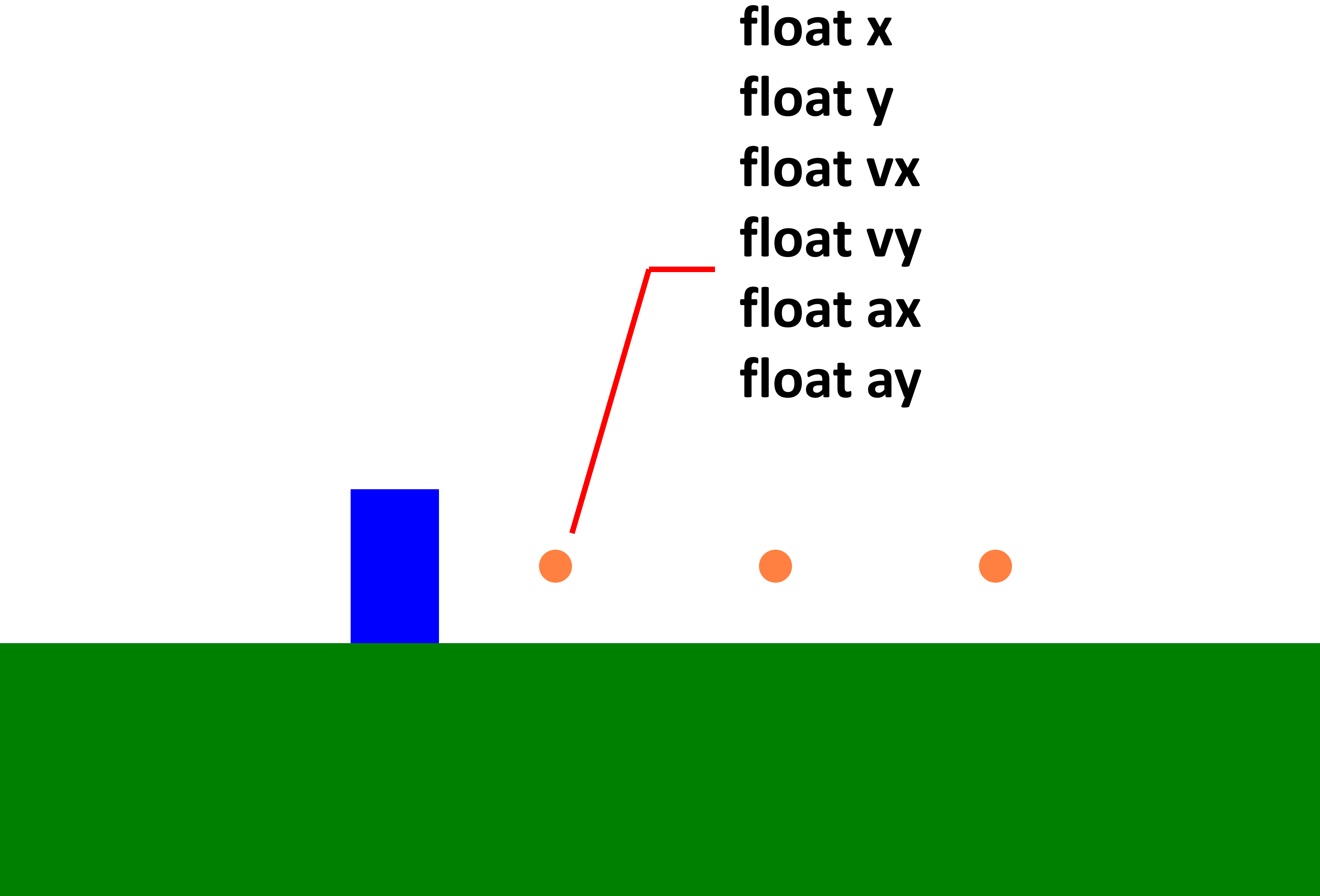
key = False

V4

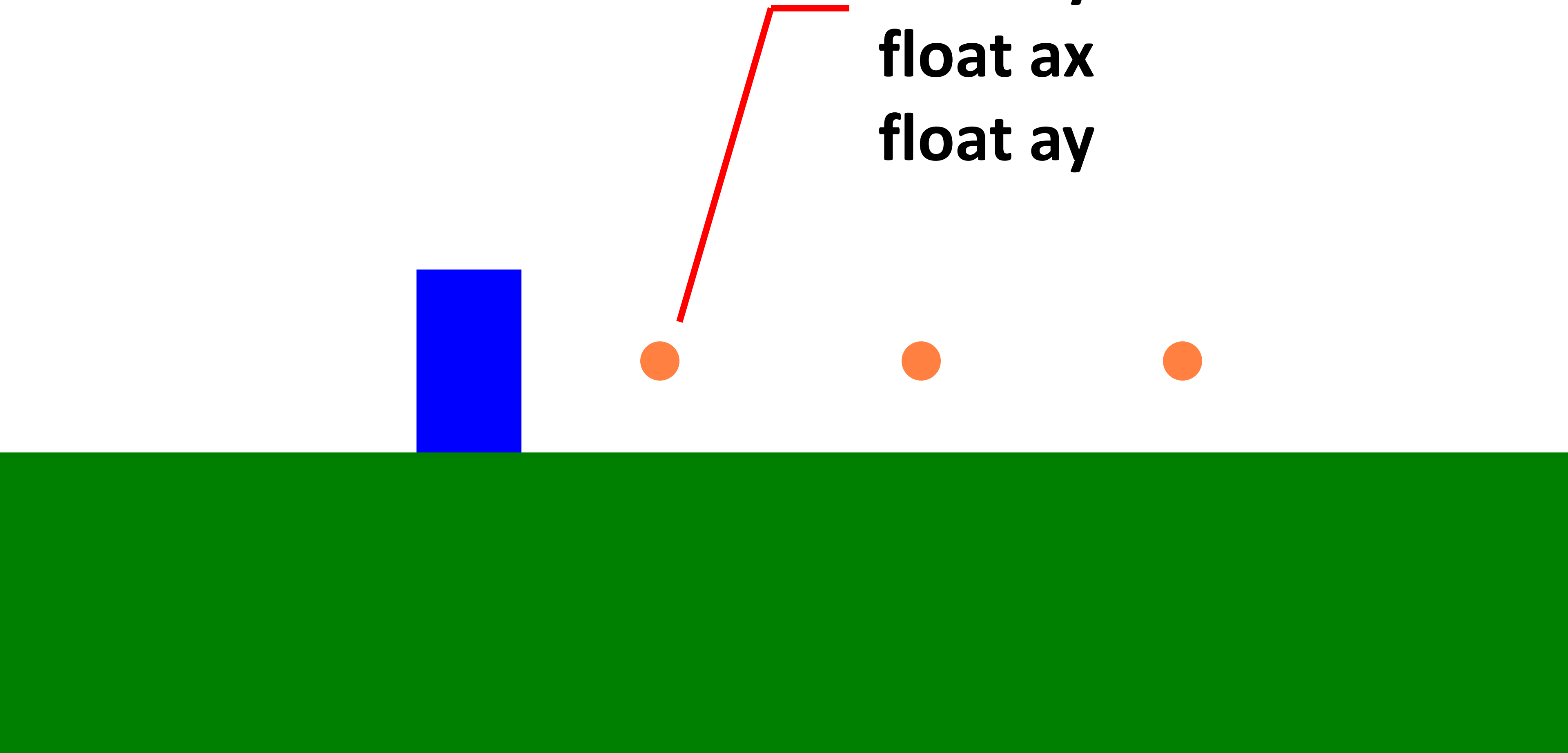
Gravity & Jumping

Bullets

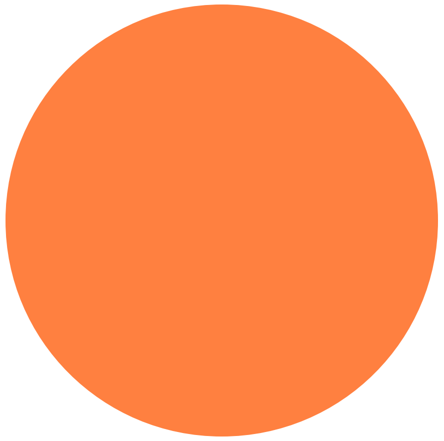




```
float x
float y
float vx
float vy
float ax
float ay
```



```
float x
float y
float vx
float vy
float ax
float ay
```



float x

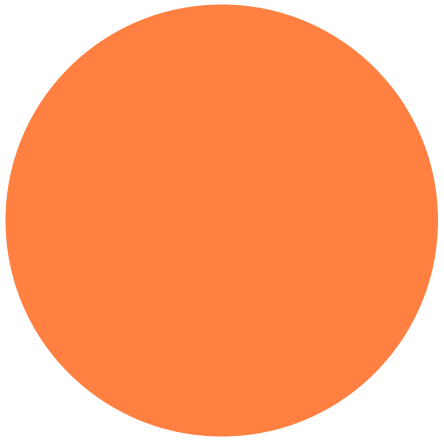
float y

float vx

float vy

float ax

float ay



```
float x
float y
float vx
float vy
float ax
float ay
```

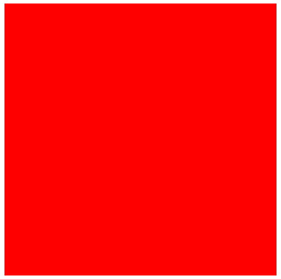
Physics:

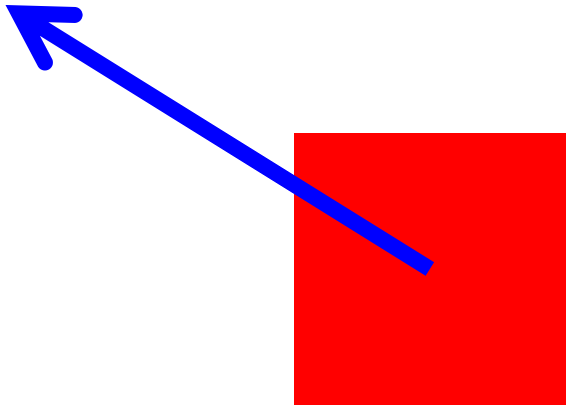
```
vx  +=  ax
vy  +=  ay
x   +=  vx
y   +=  vy
```

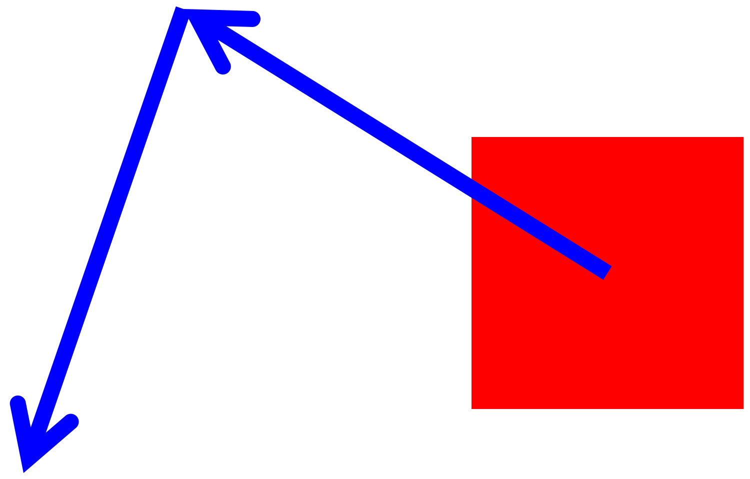
every frame

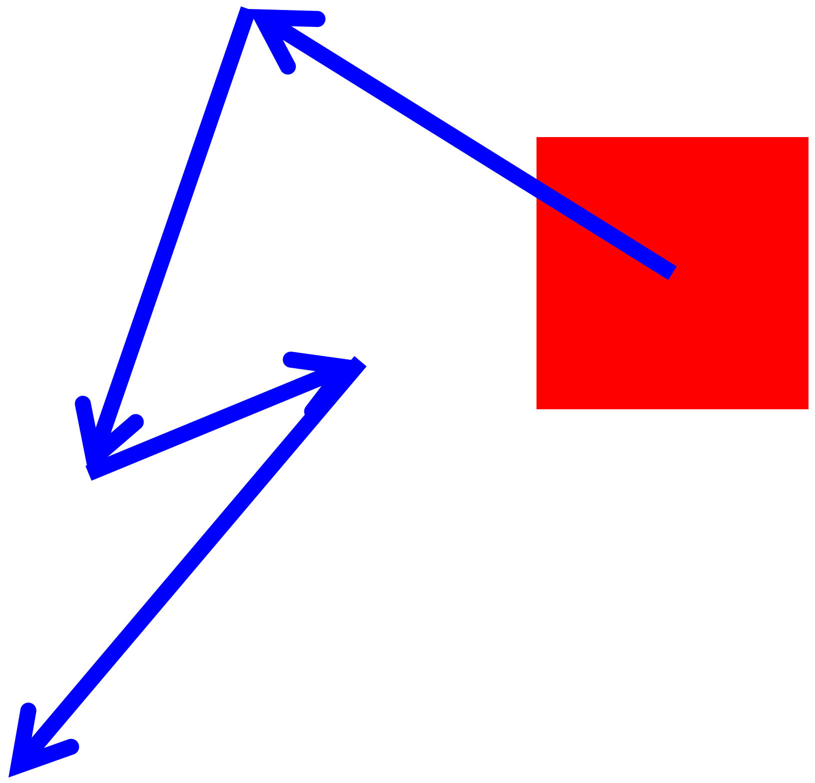
V5

Bullets





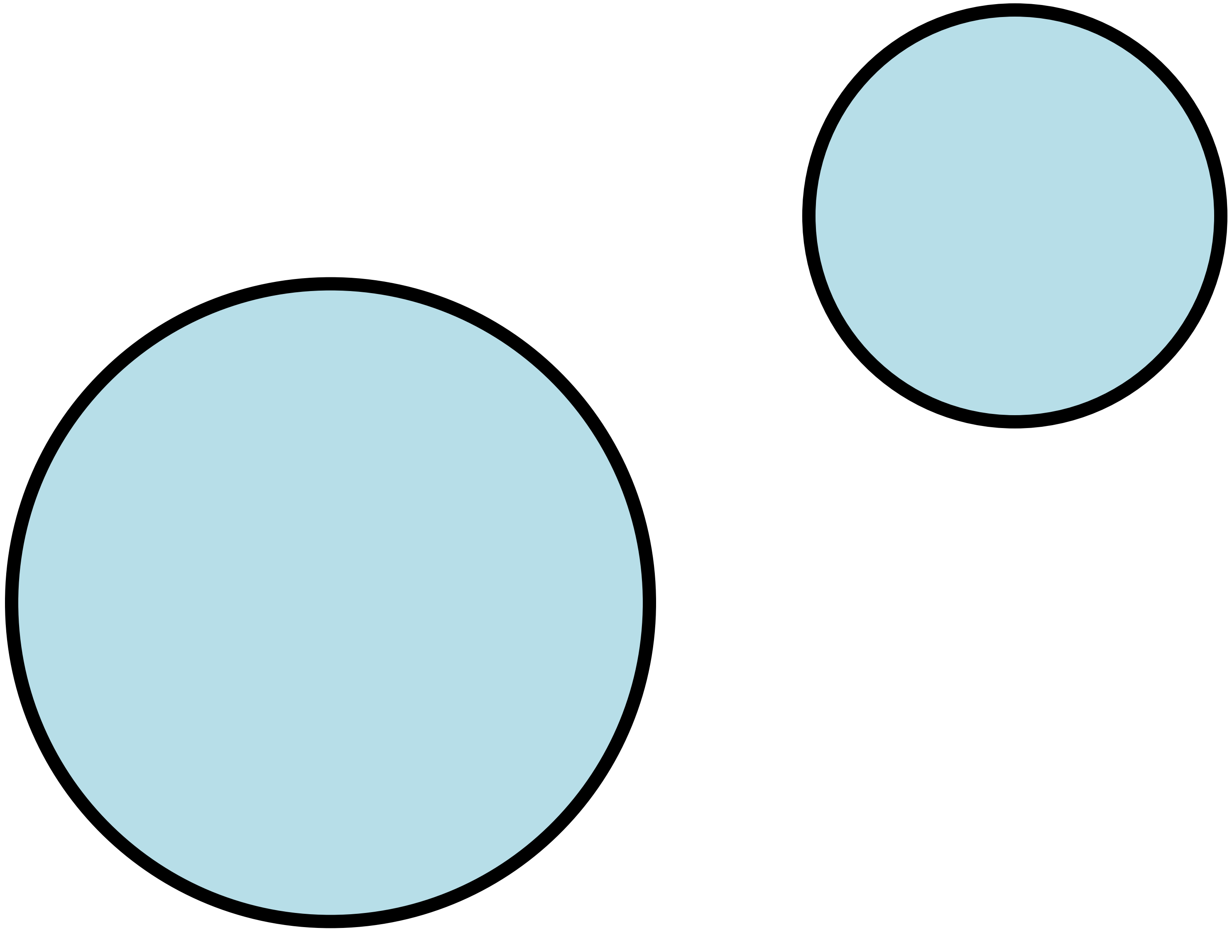


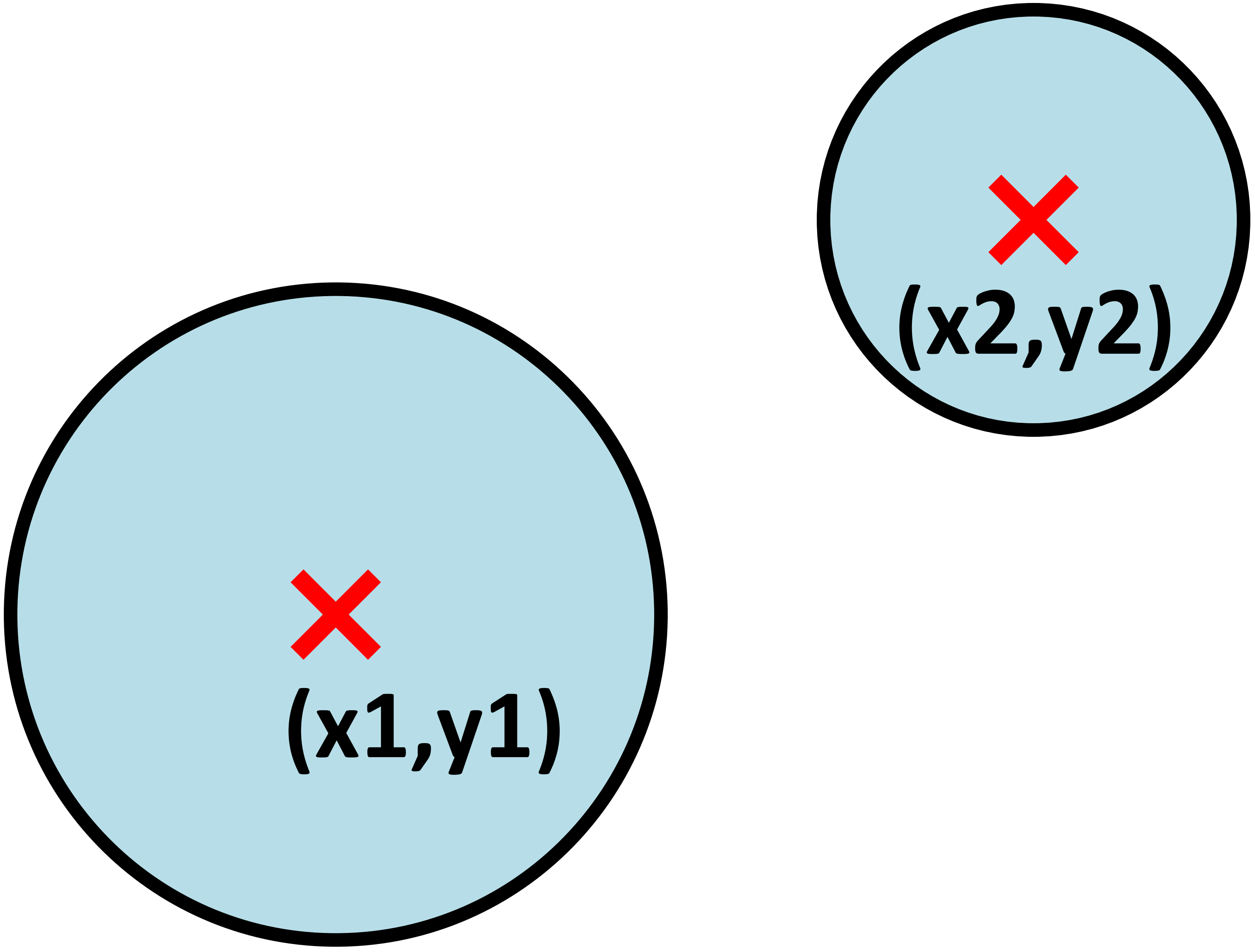


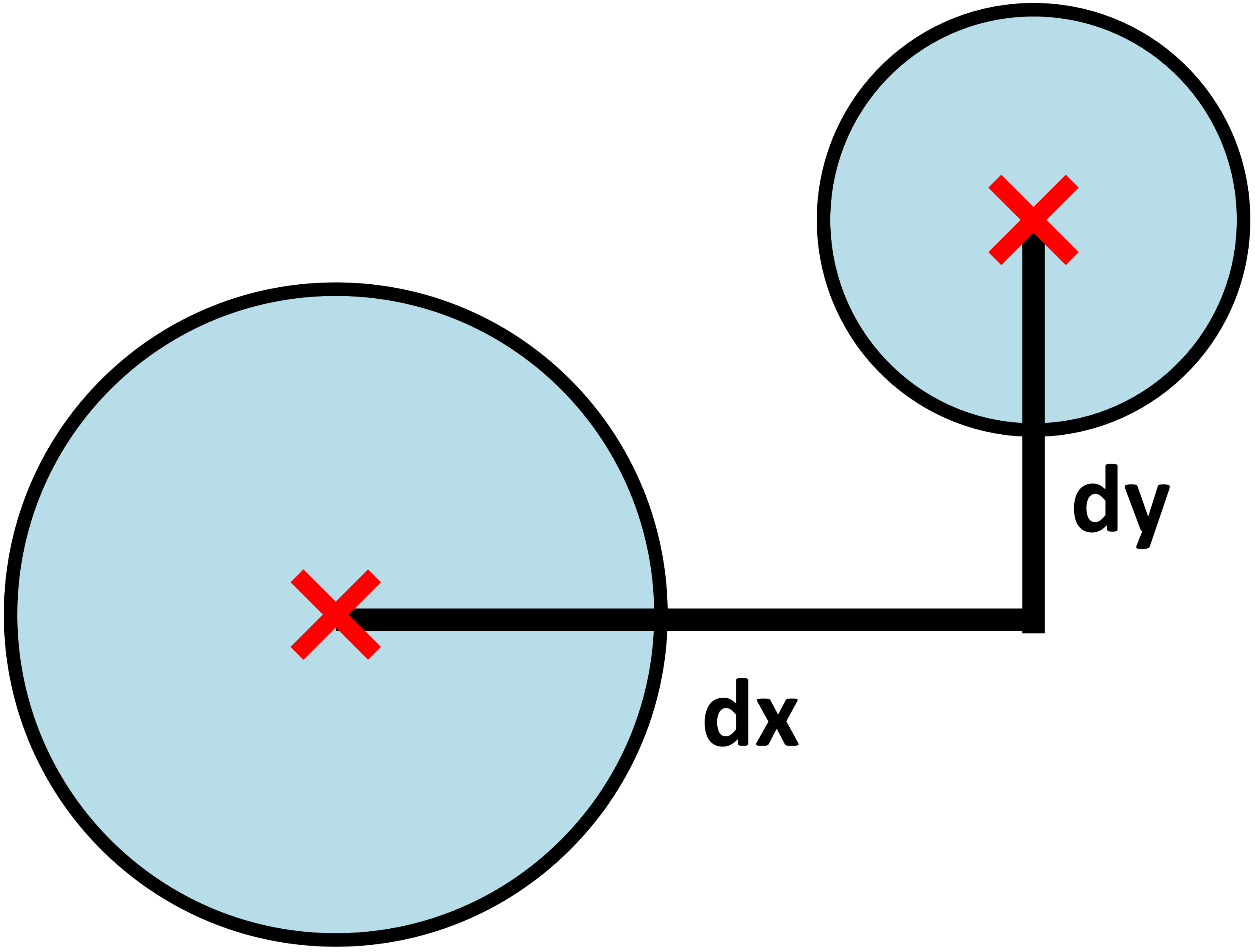
V6

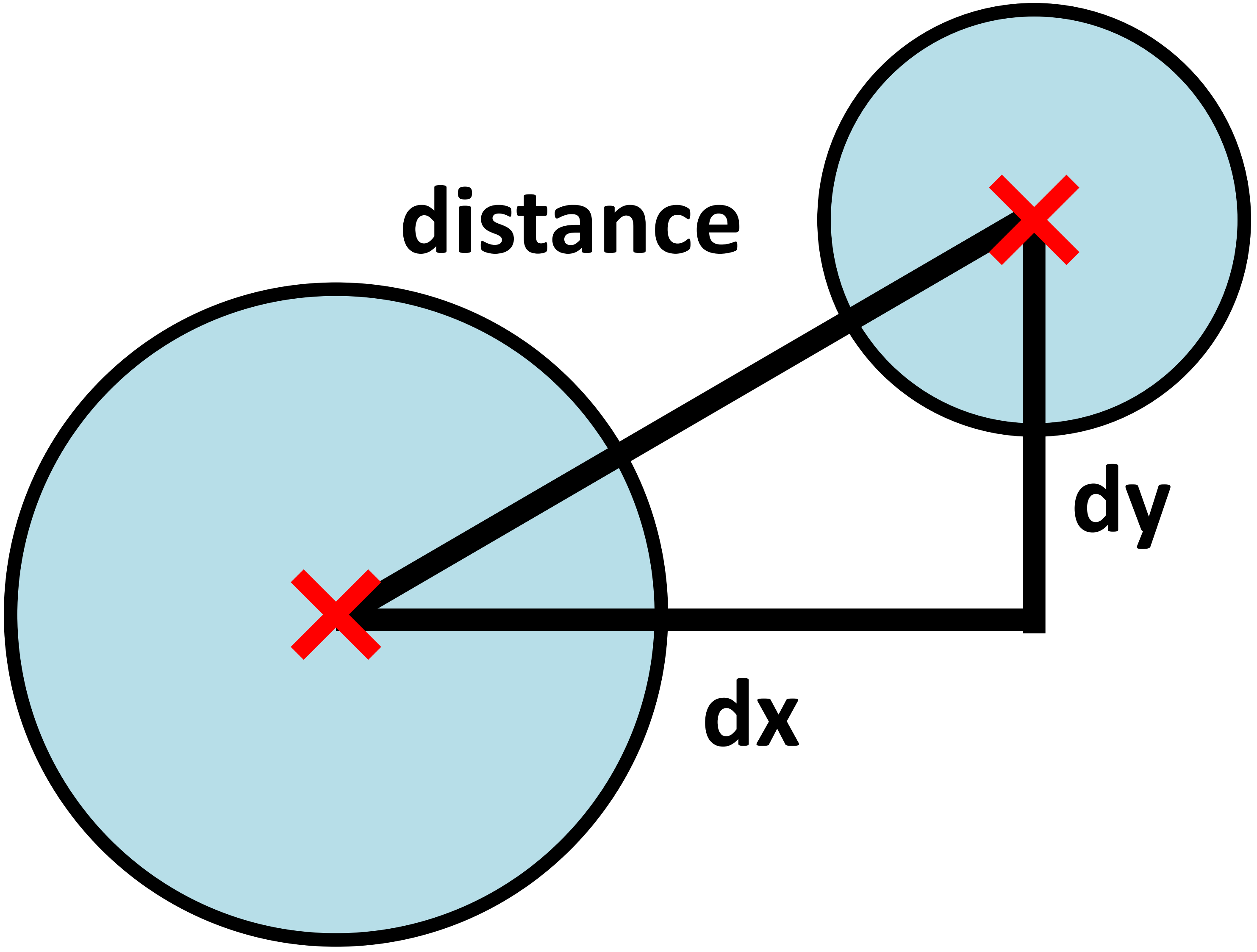
Enemies

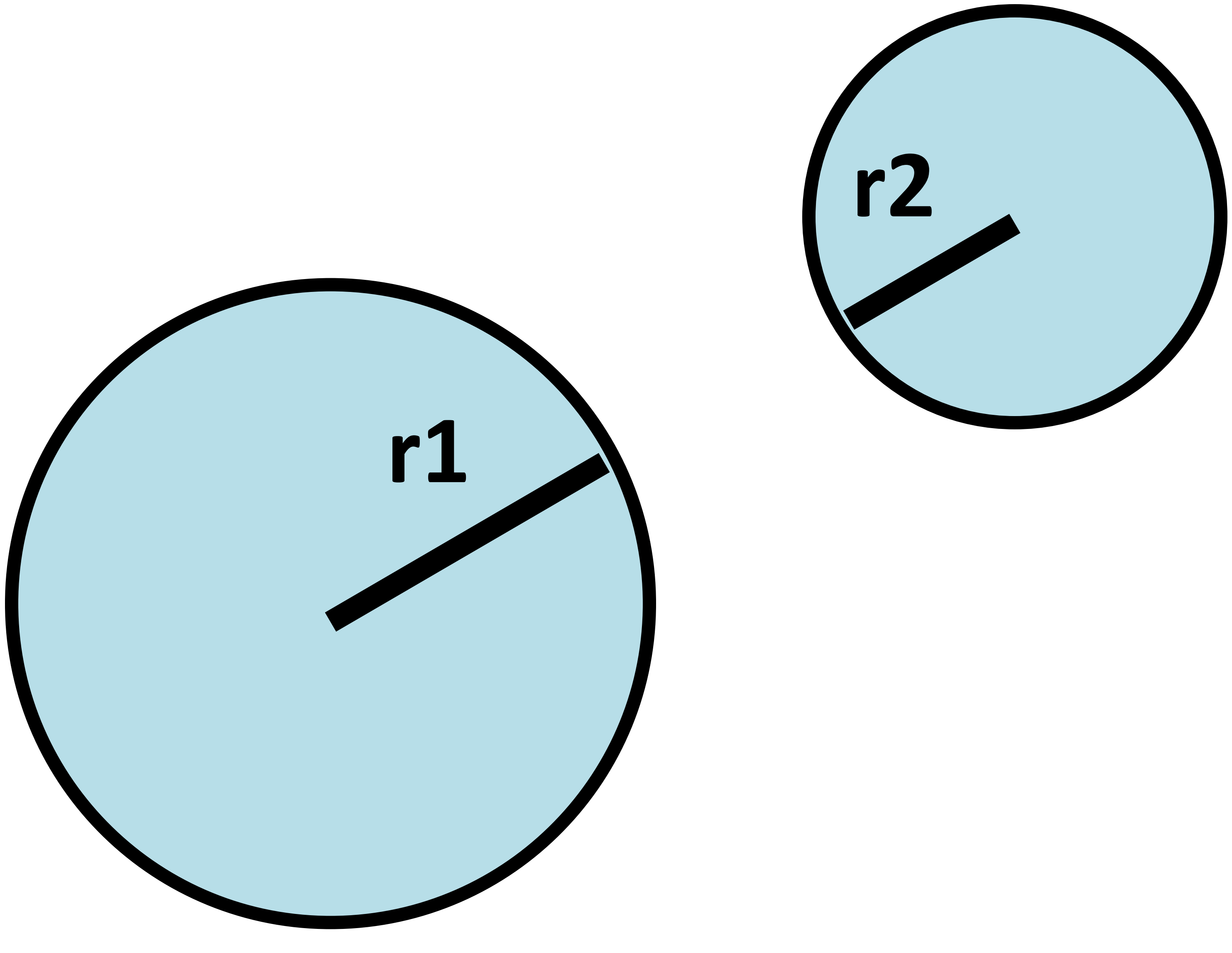
Collision: Circle on Circle

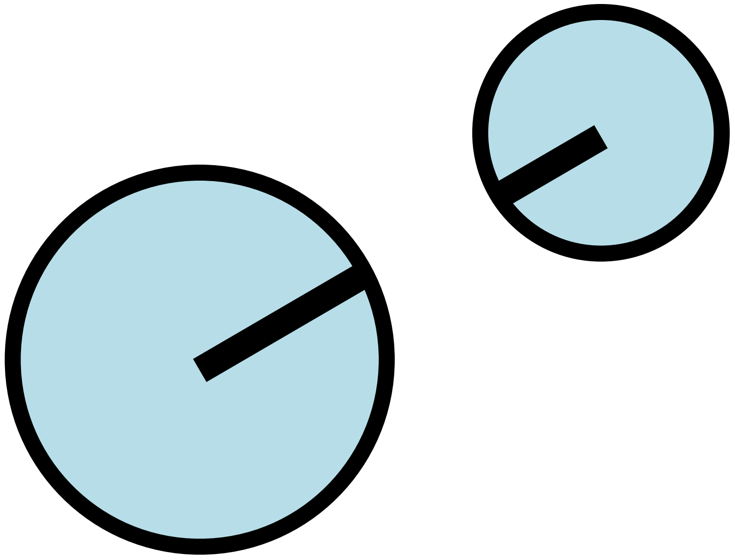


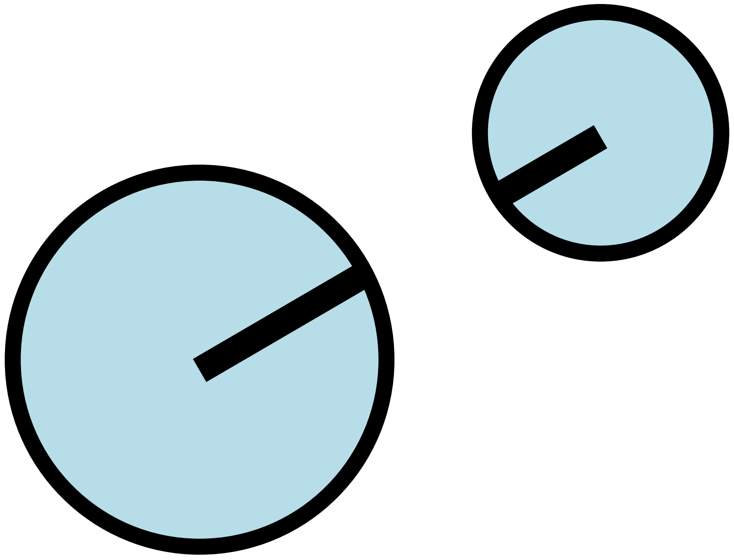

(x1,y1)
(x2,y2)


dy
dx

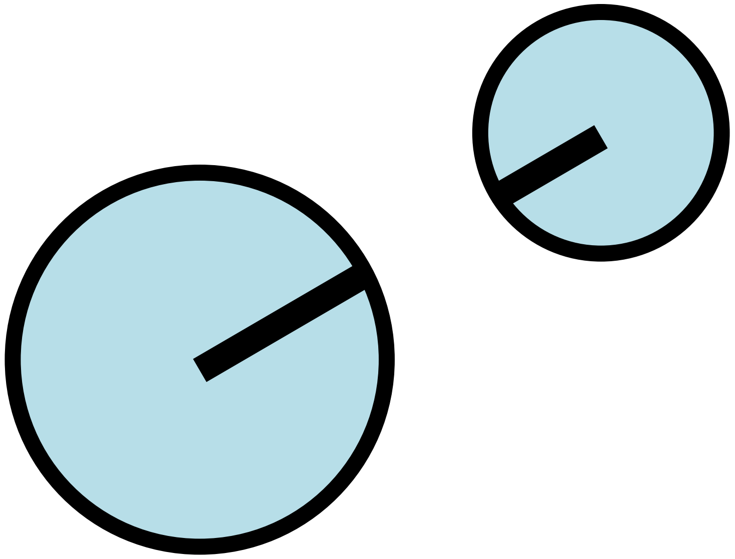

distance
dy
dx


r1
r2

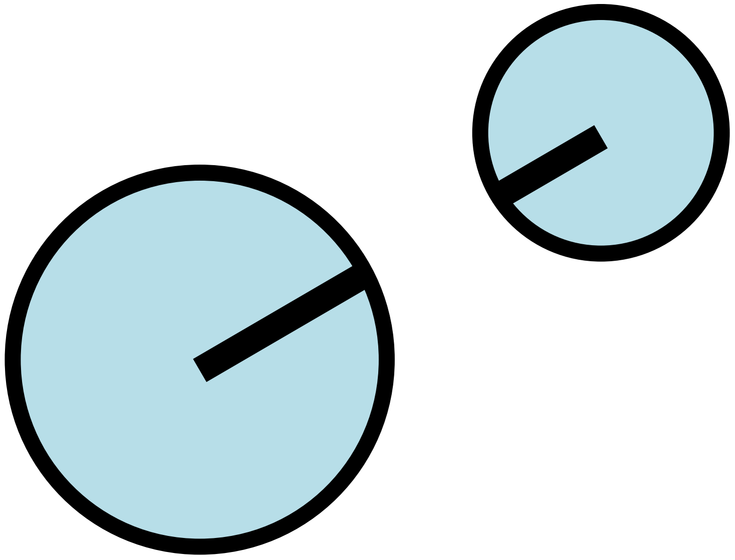




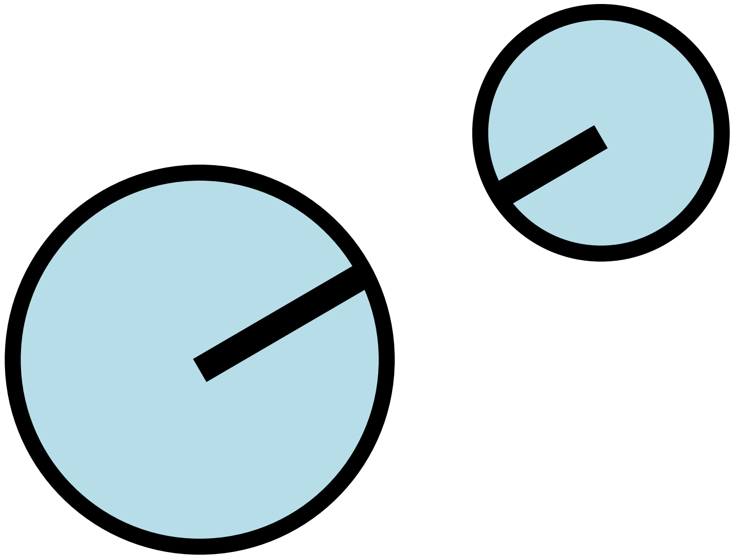
dx = x1 – x2



$$dx = x1 - x2$$
$$dy = y1 - y2$$



$$dx = x1 - x2$$
$$dy = y1 - y2$$
$$r = r1 + r2$$



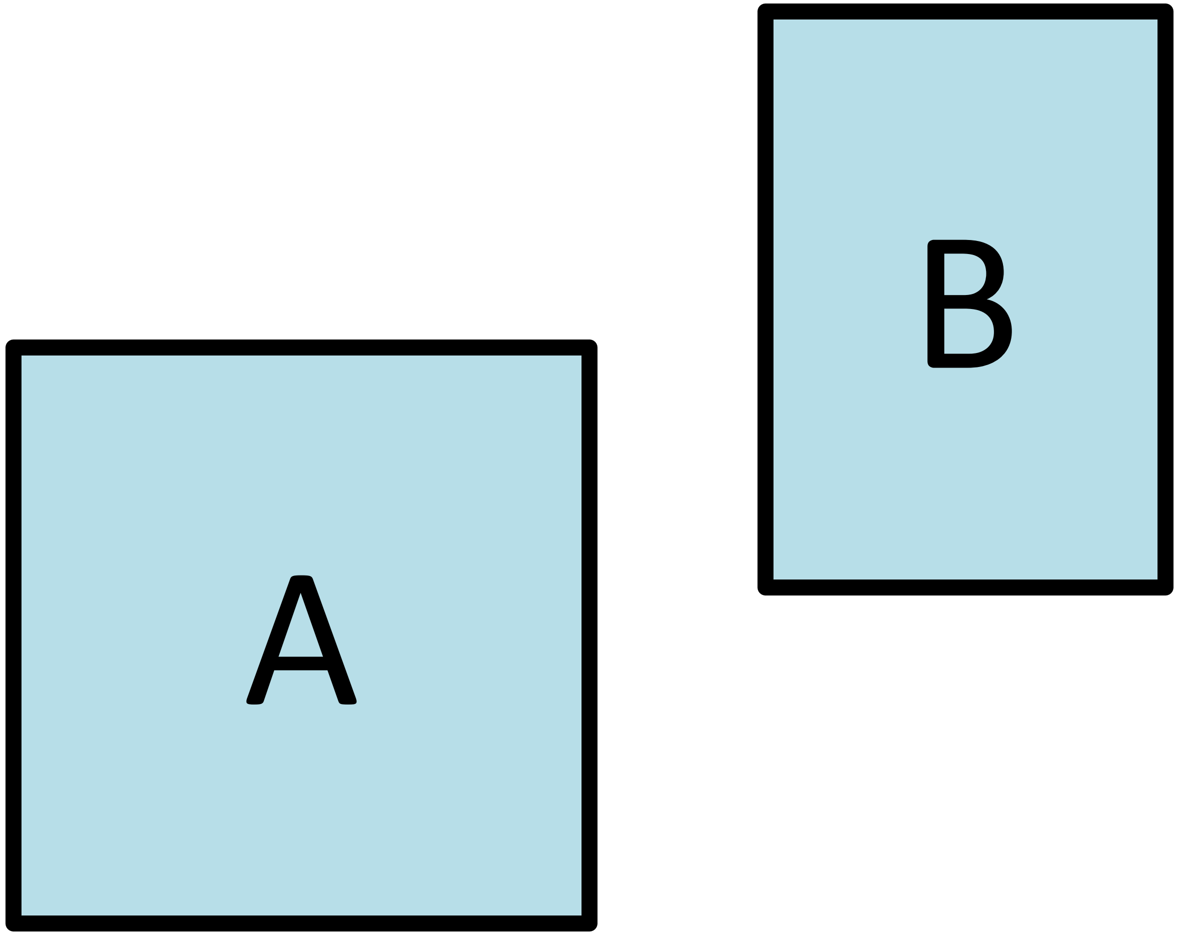
dx = x1 – x2

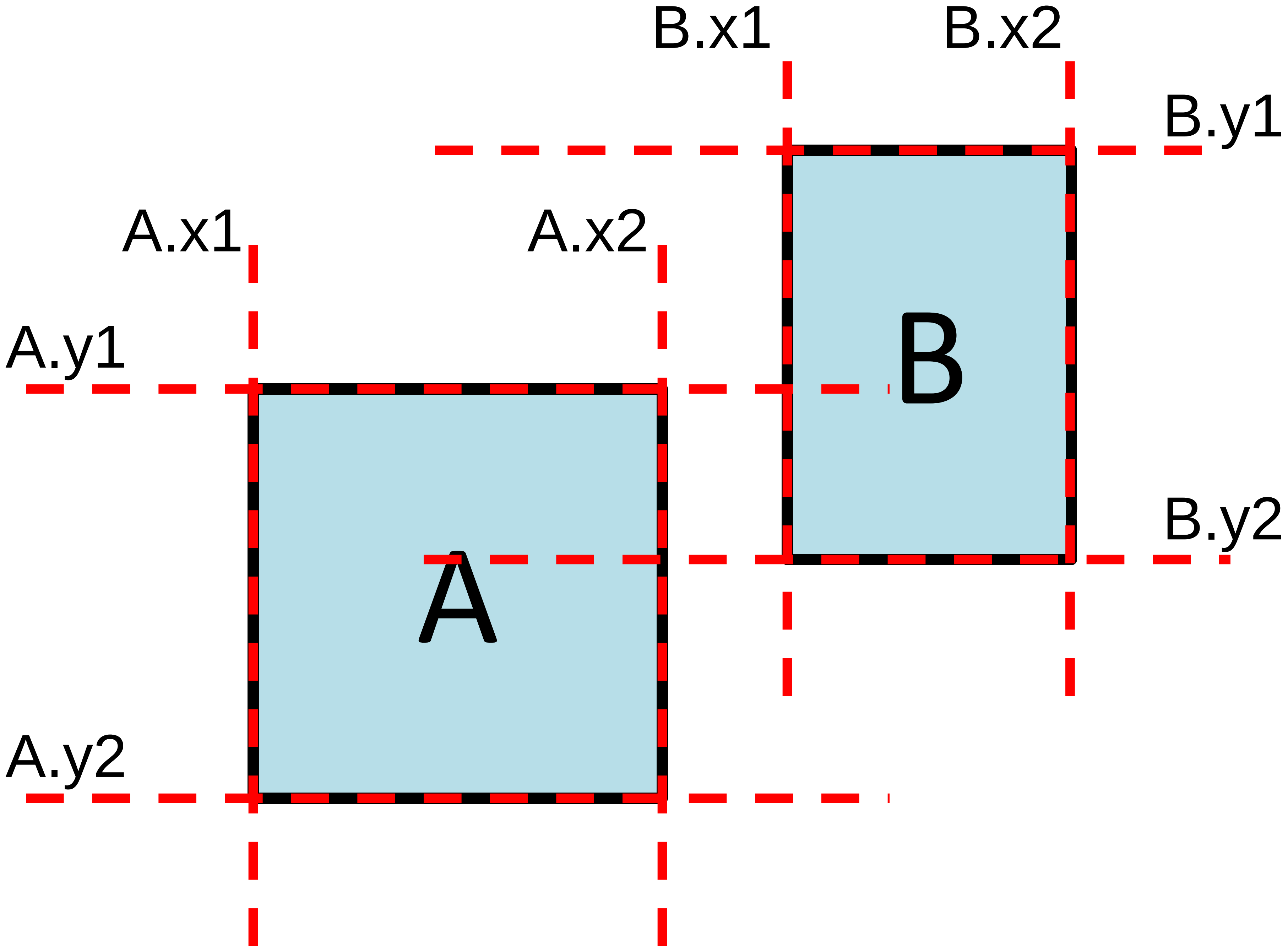
dy = y1 – y2

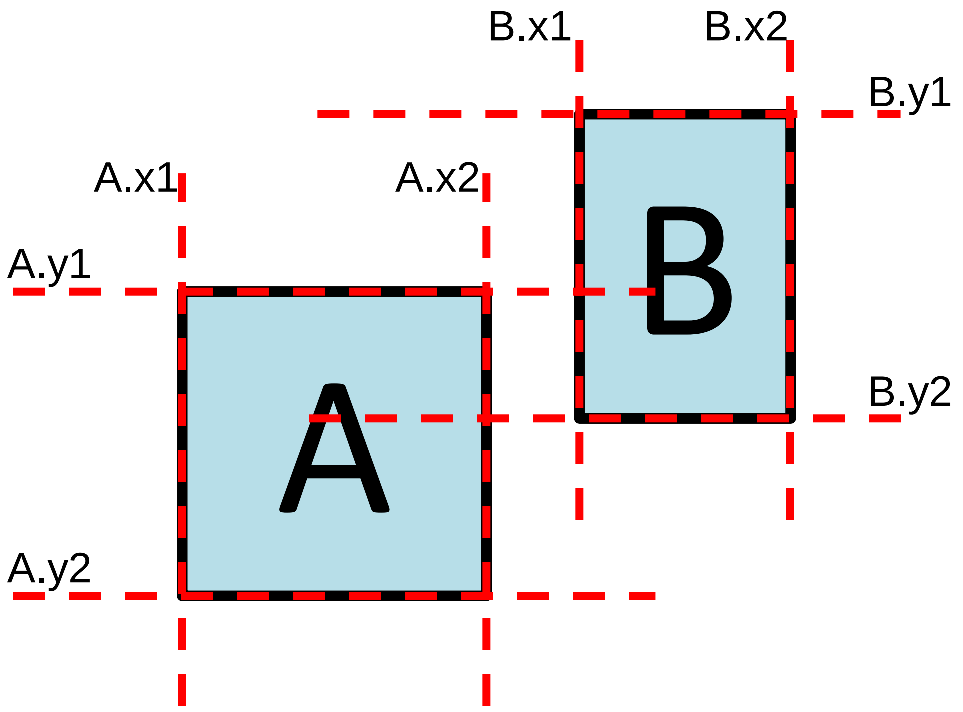
r = r1 + r2

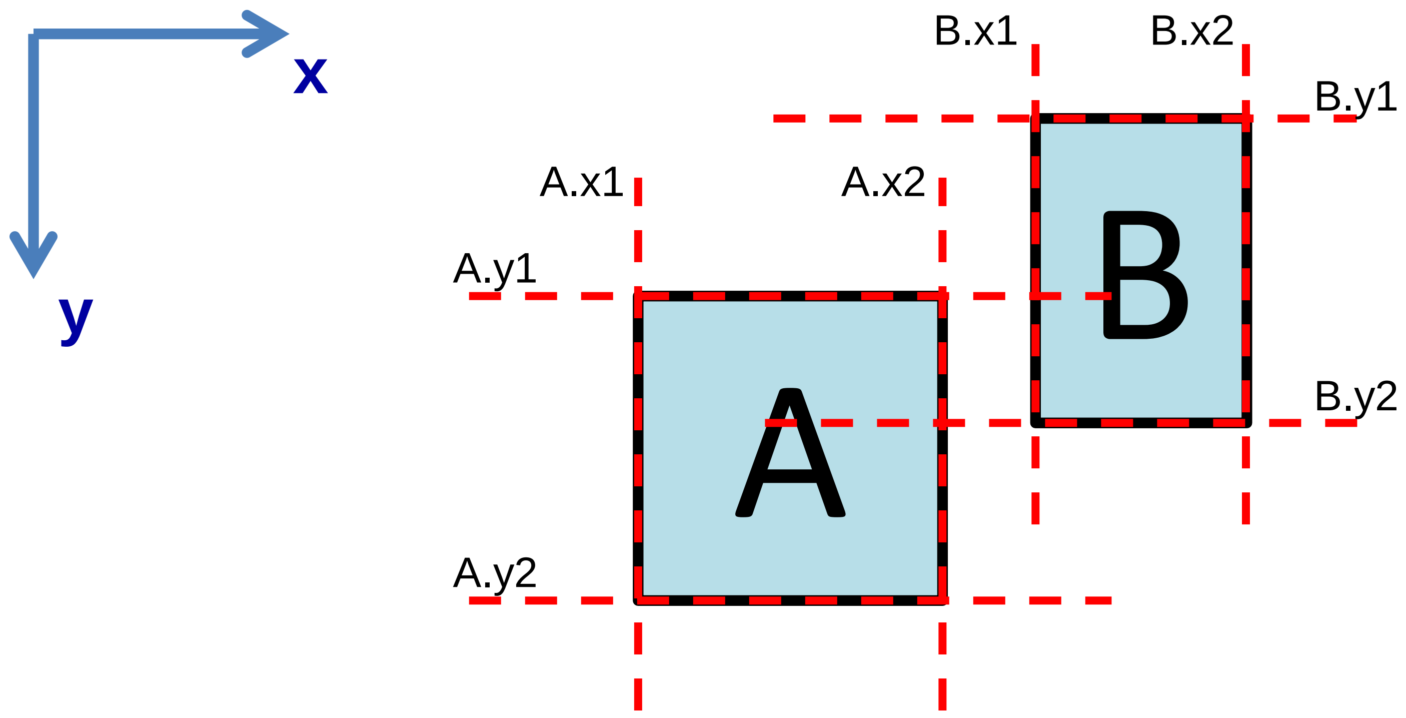
dx*dx + dy*dy <= r*r

Collision:
Rectangle on Rectangle (axis-aligned)

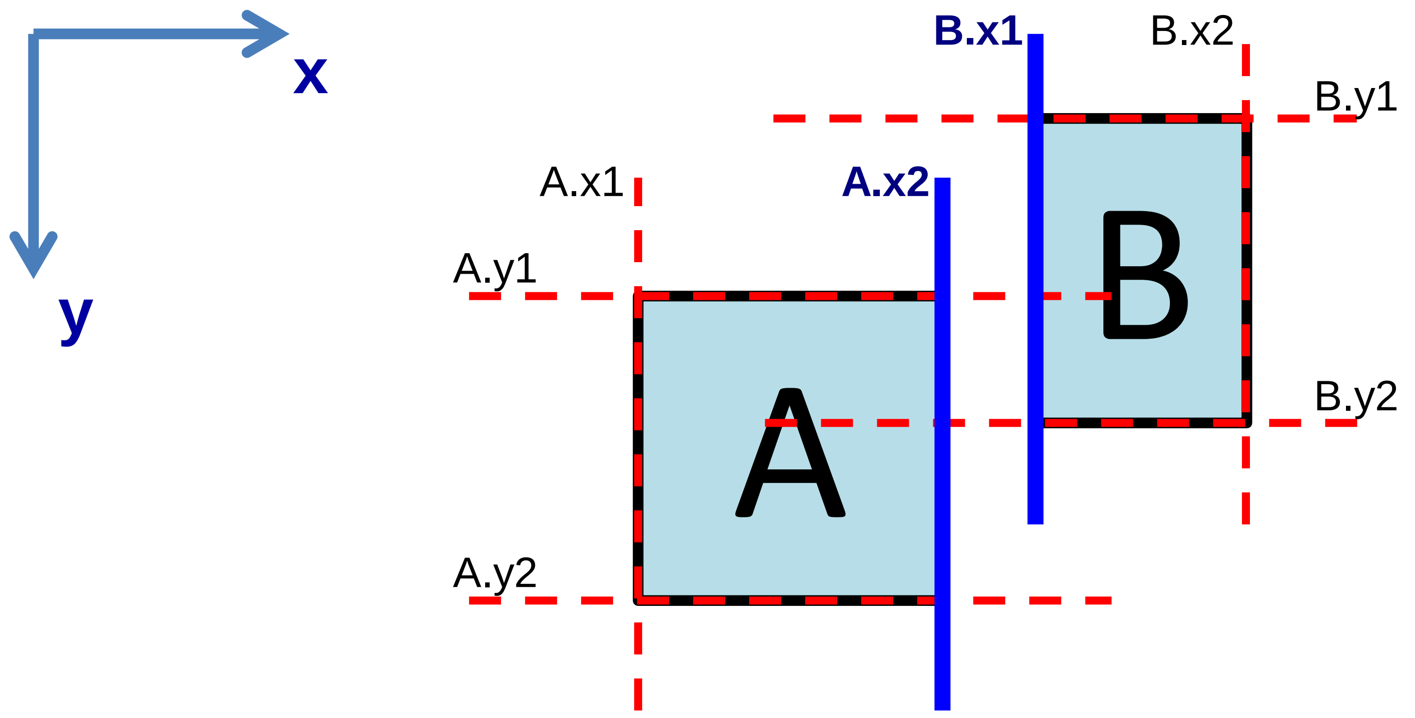

A
B


B.x1
B.x2
B.y1
A.x1
A.x2
A.y1
B
B.y2
A
A.y2


B.x1
B.x2
B.y1
A.x1
A.x2
A.y1
B
B.y2
A
A.y2

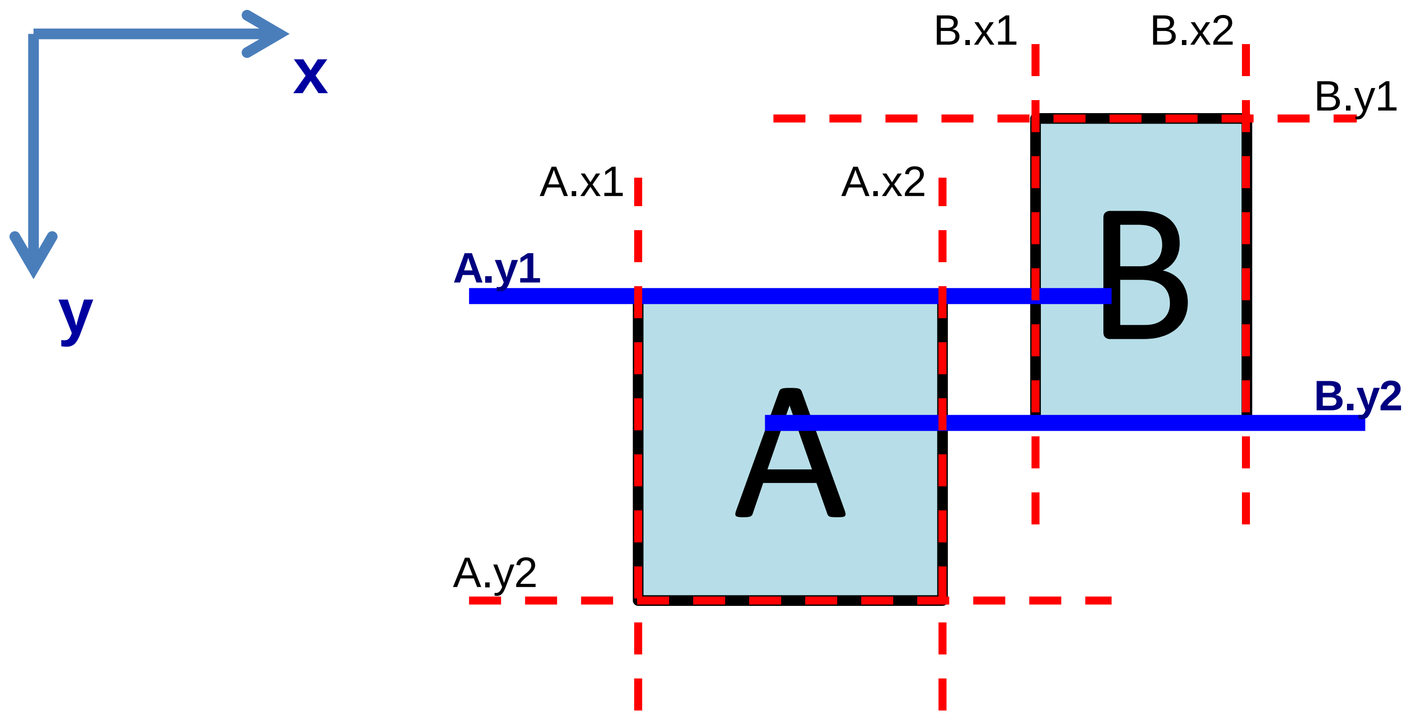


A and B don’t collide if:



A and B don’t collide if:

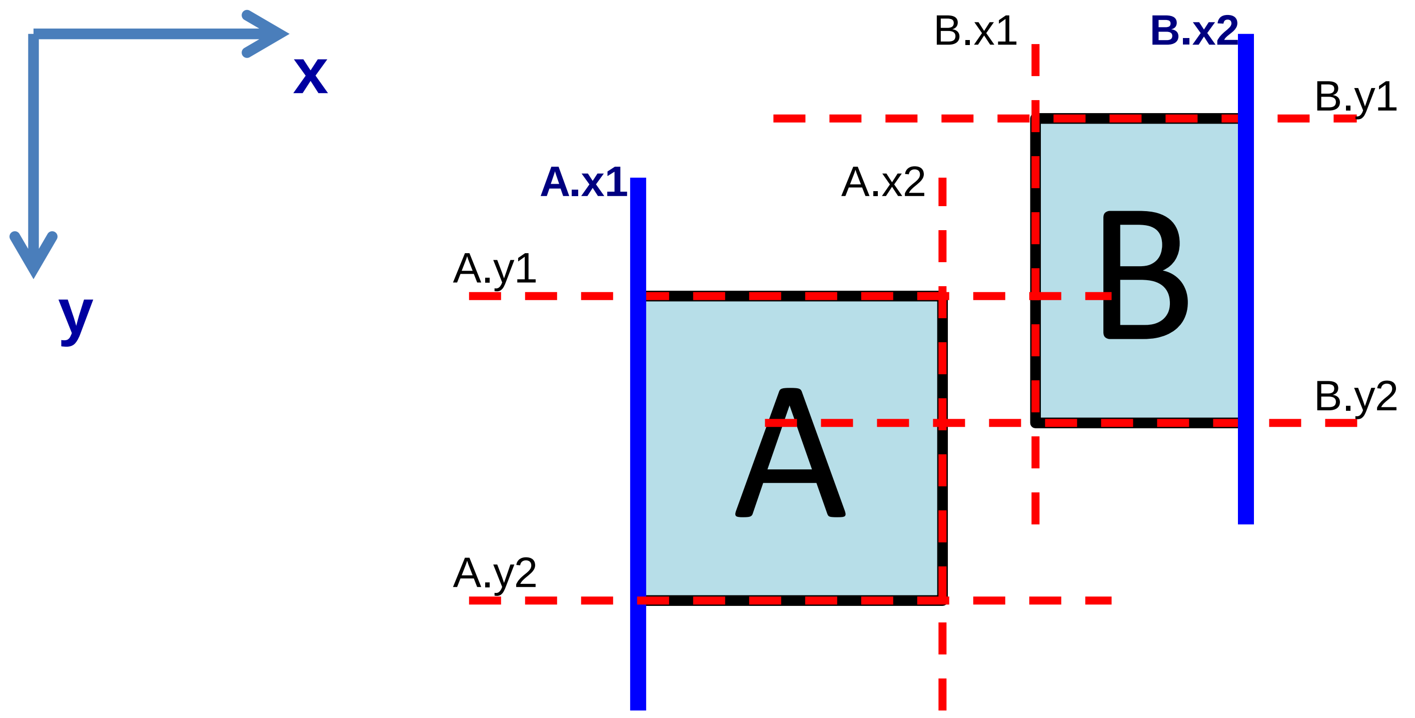
A.x2 < B.x1



A and B don't collide if:

A.x2 < B.x1 OR

B.y2 < A.y1

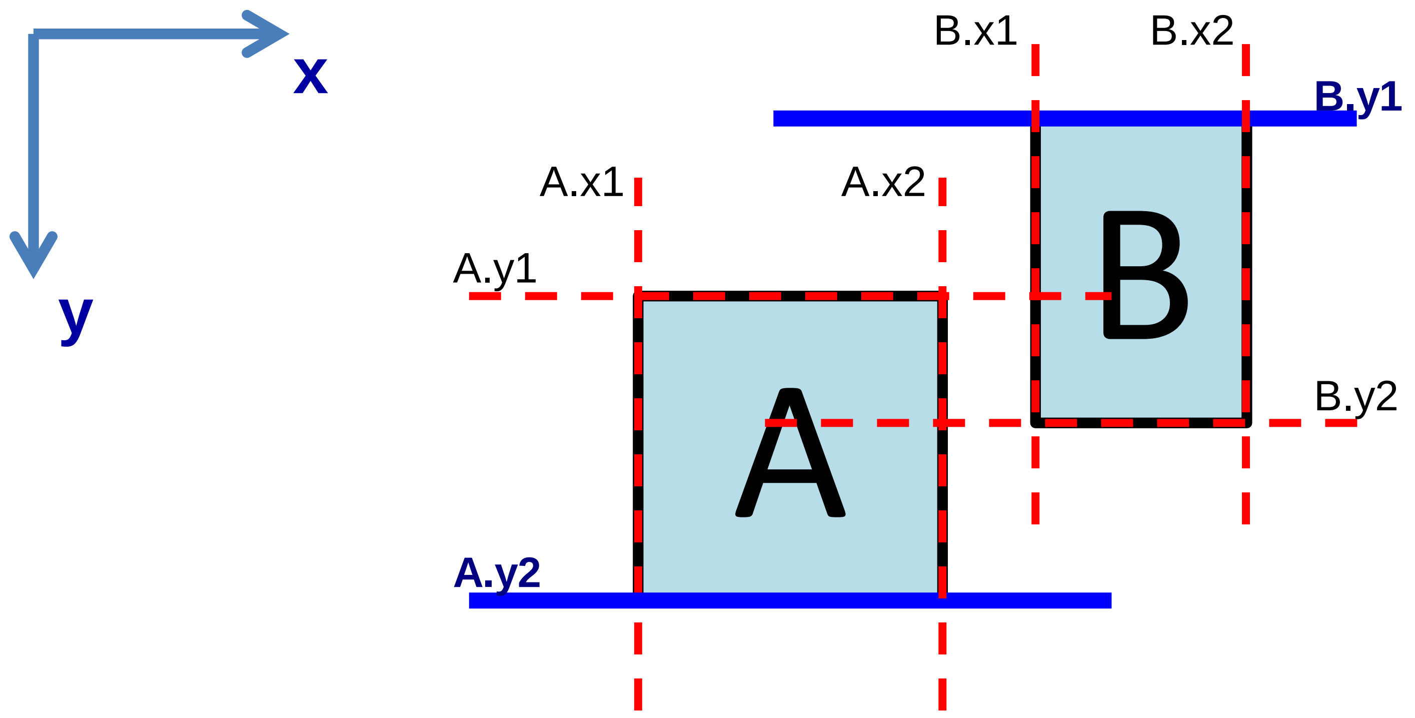


A and B don’t collide if:

A.x2 < B.x1 OR

B.y2 < A.y1 OR

B.x2 < A.x1



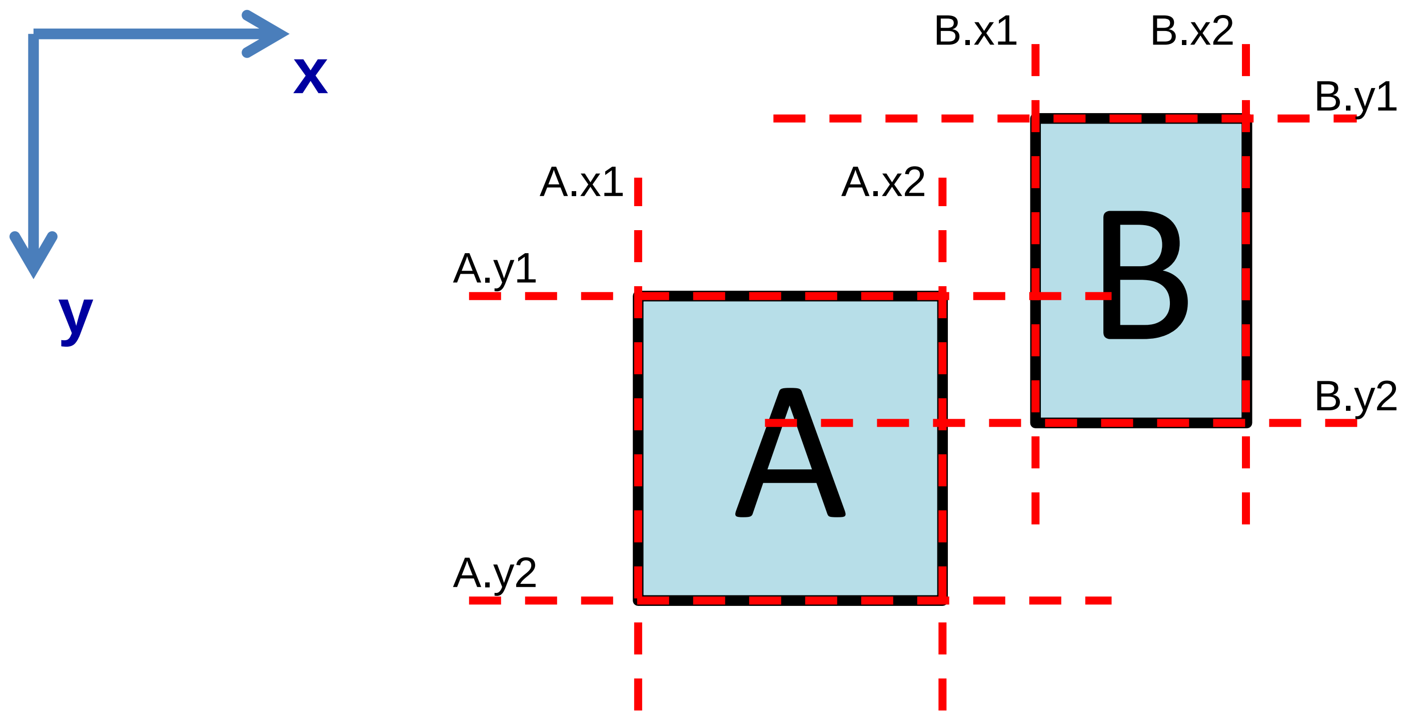
A and B don't collide if:

A.x2 < B.x1 OR

B.y2 < A.y1 OR

B.x2 < A.x1 OR

A.y2 < B.y1



A and B don’t collide if:

A.x2 < B.x1 **OR**

B.y2 < A.y1 **OR**

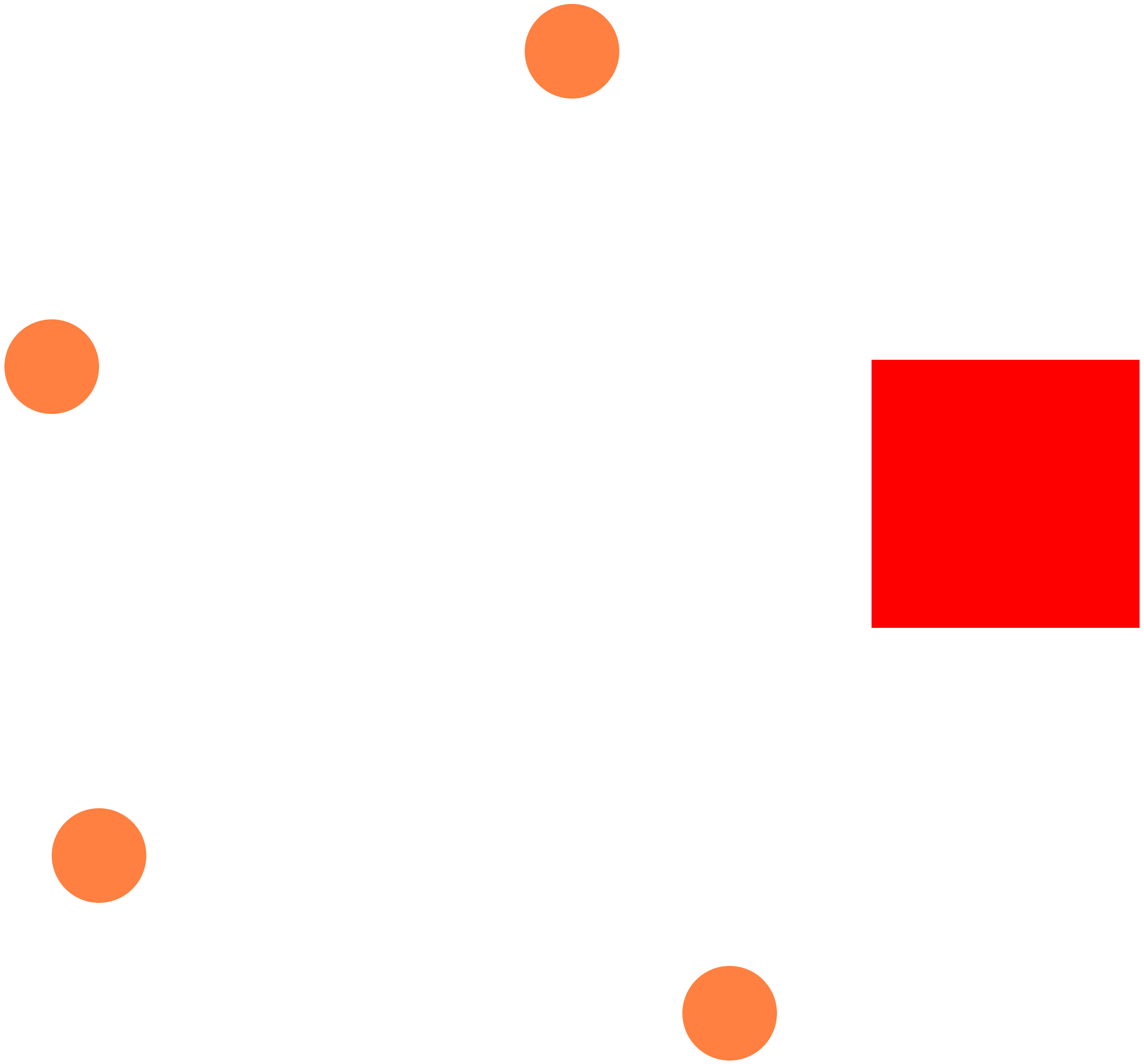
B.x2 < A.x1 **OR**

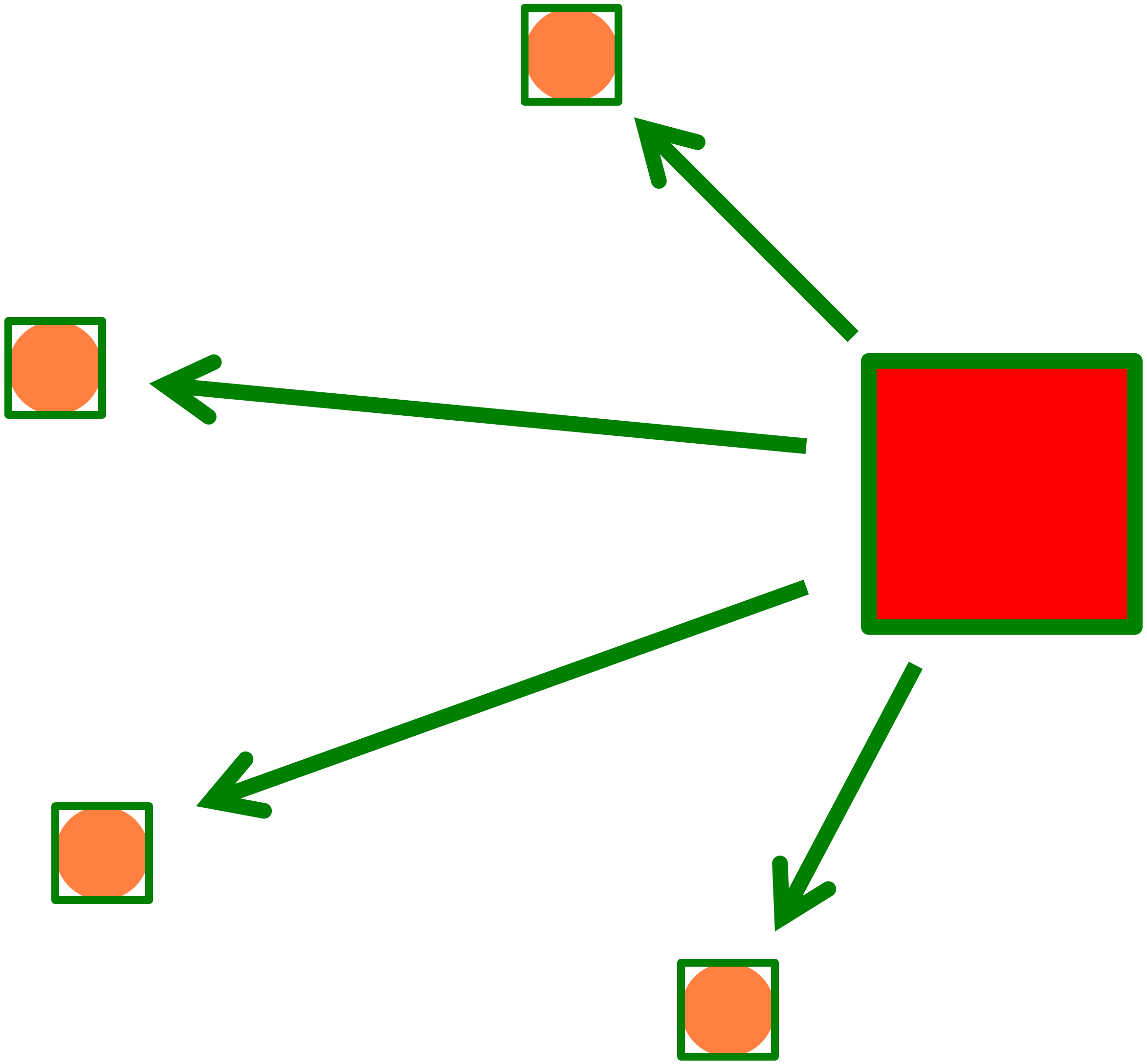
A.y2 < B.y1

} Otherwise, they collide!

Collision:

Implementation





V7

Collision Detection

V8

Break – TODOs:

1. Enemy movement
2. Enemies that can actually hurt you

Platforms

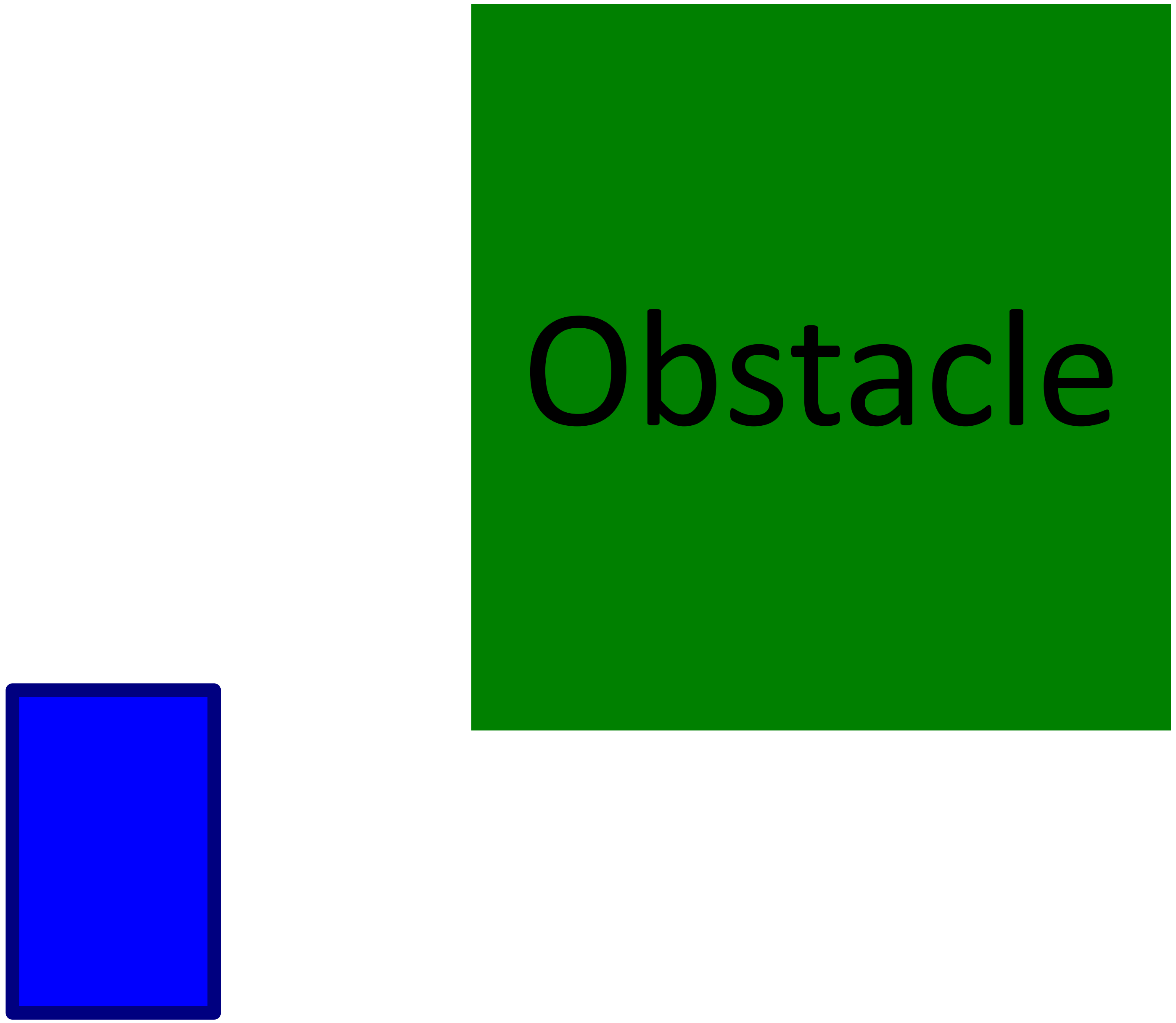
(Not easy!)

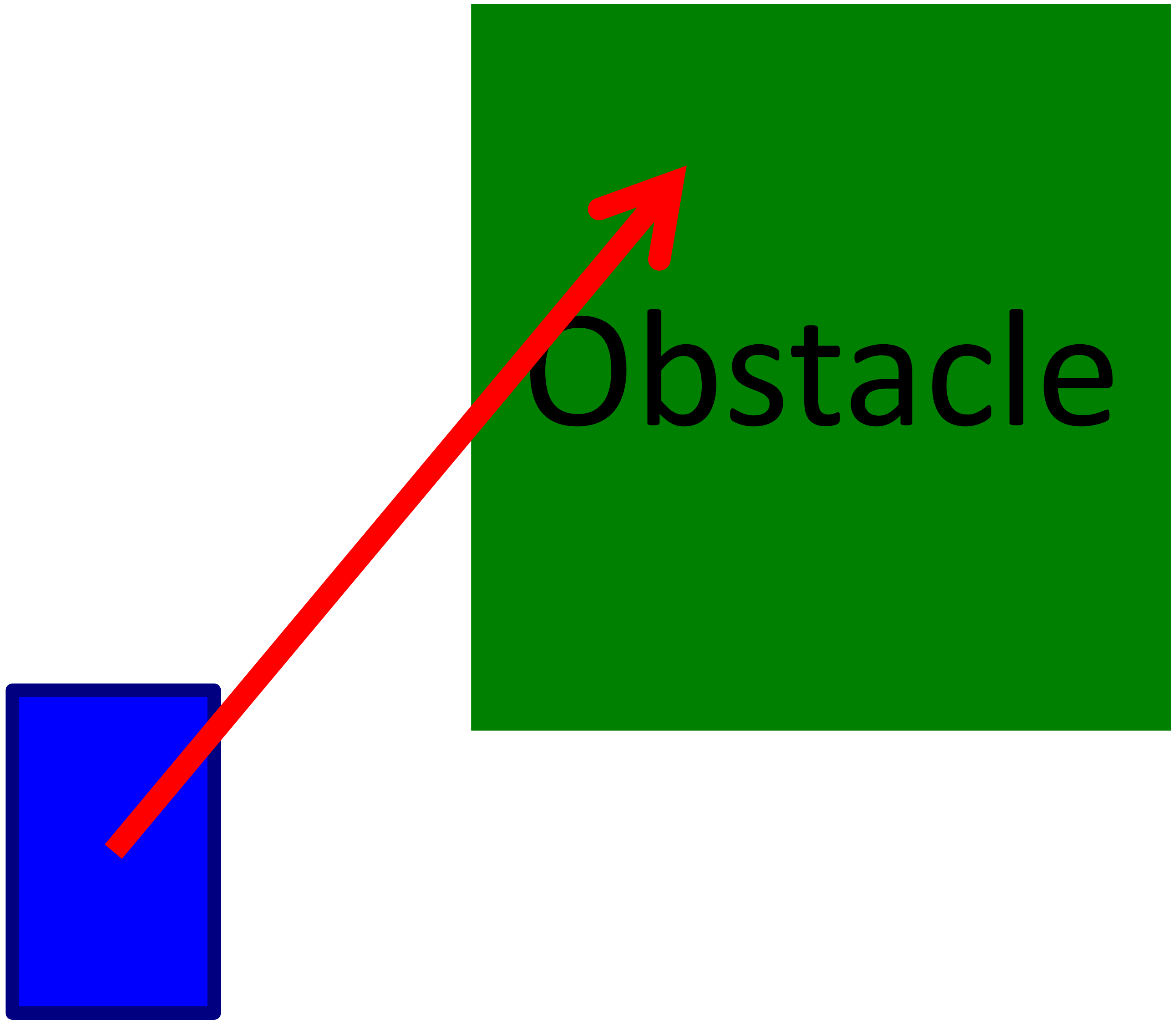
Two Parts of Collision

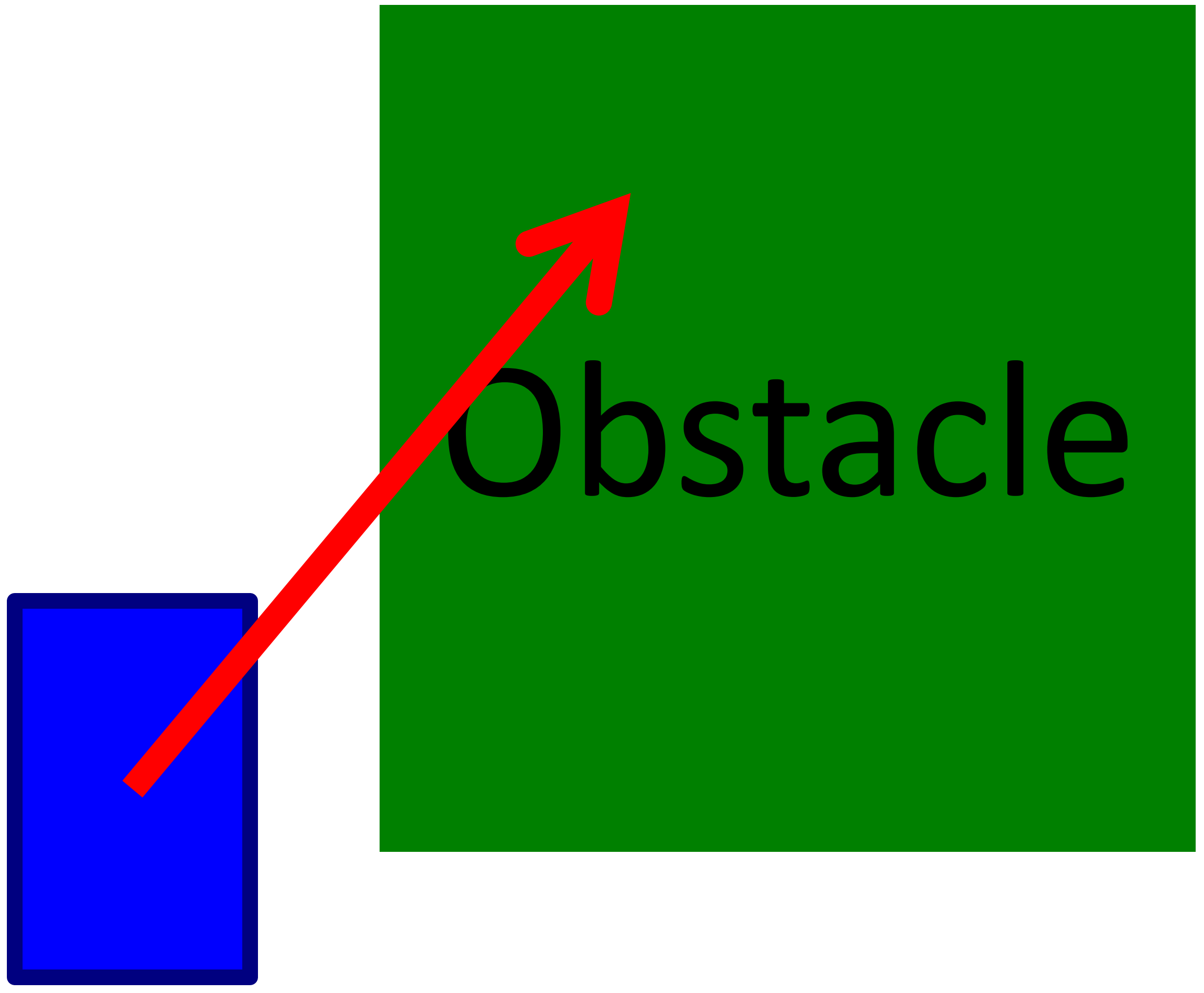
1. How do you detect collision?
2. What do you do upon collision?

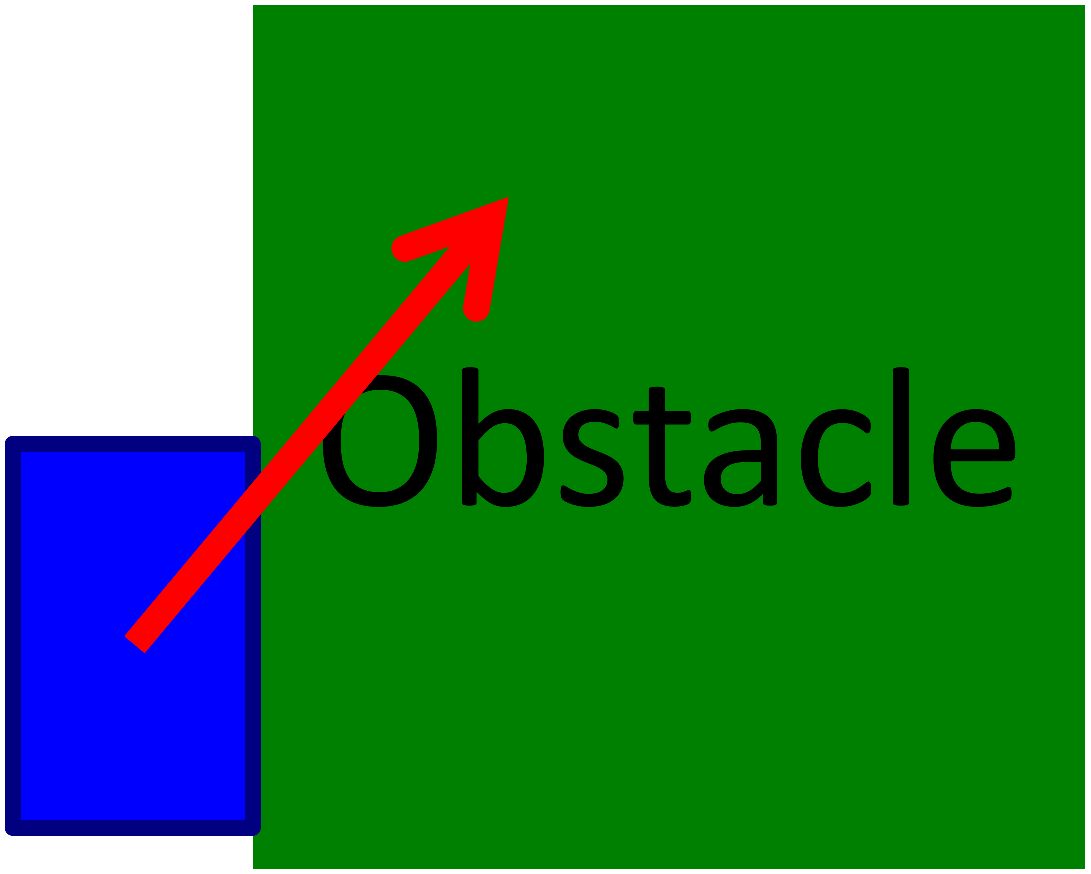
1. How do you detect collision?
2. What do you do upon collision?

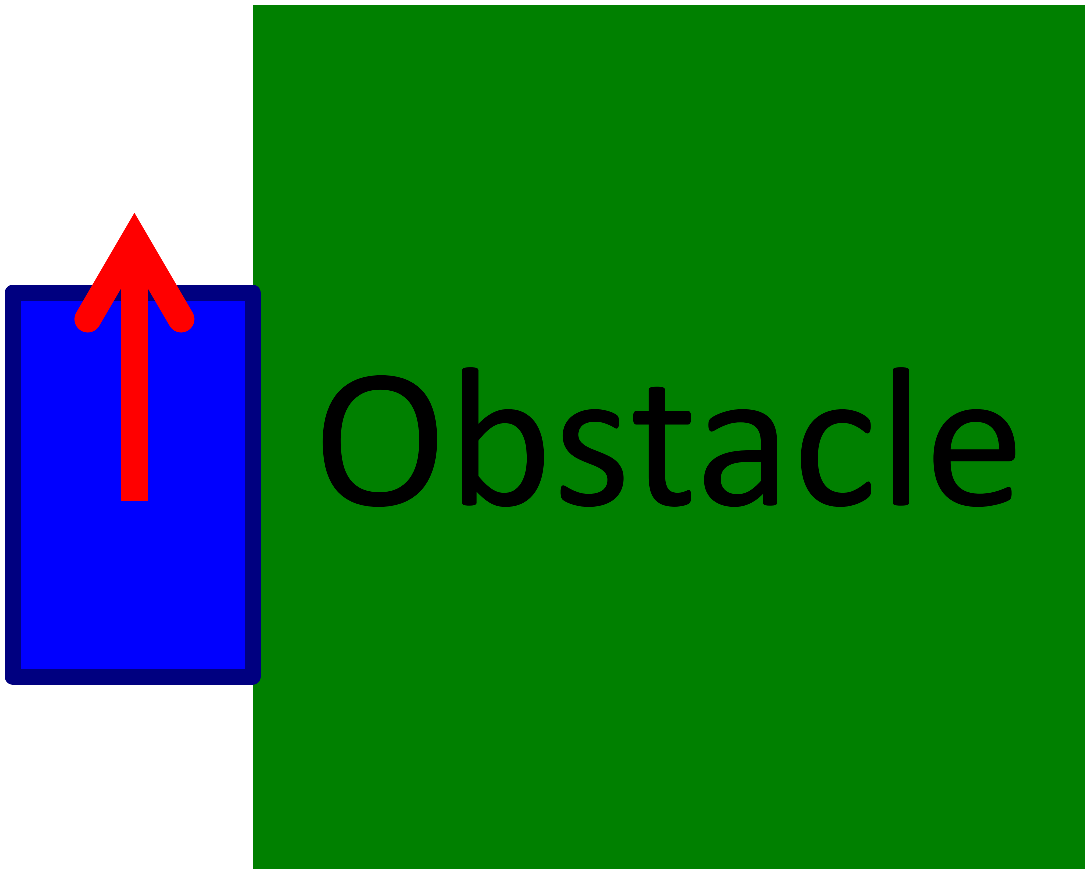
Obstacle

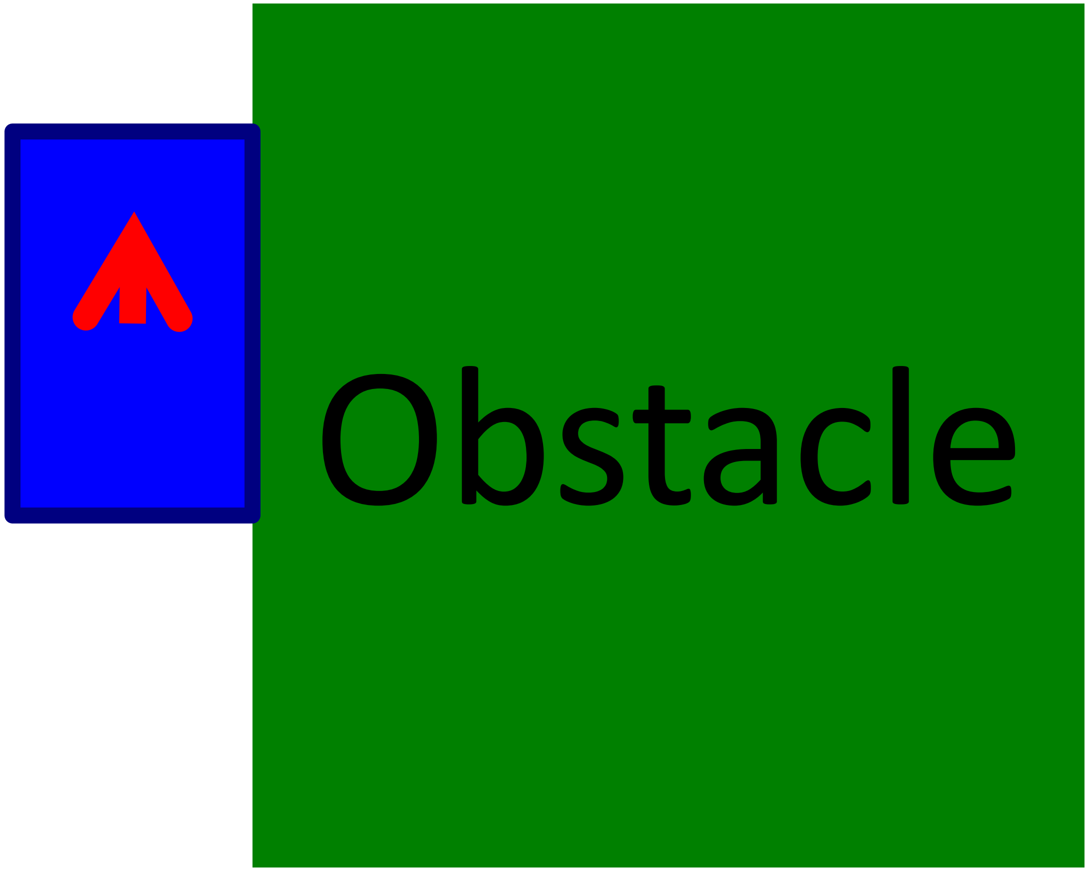

Obstacle

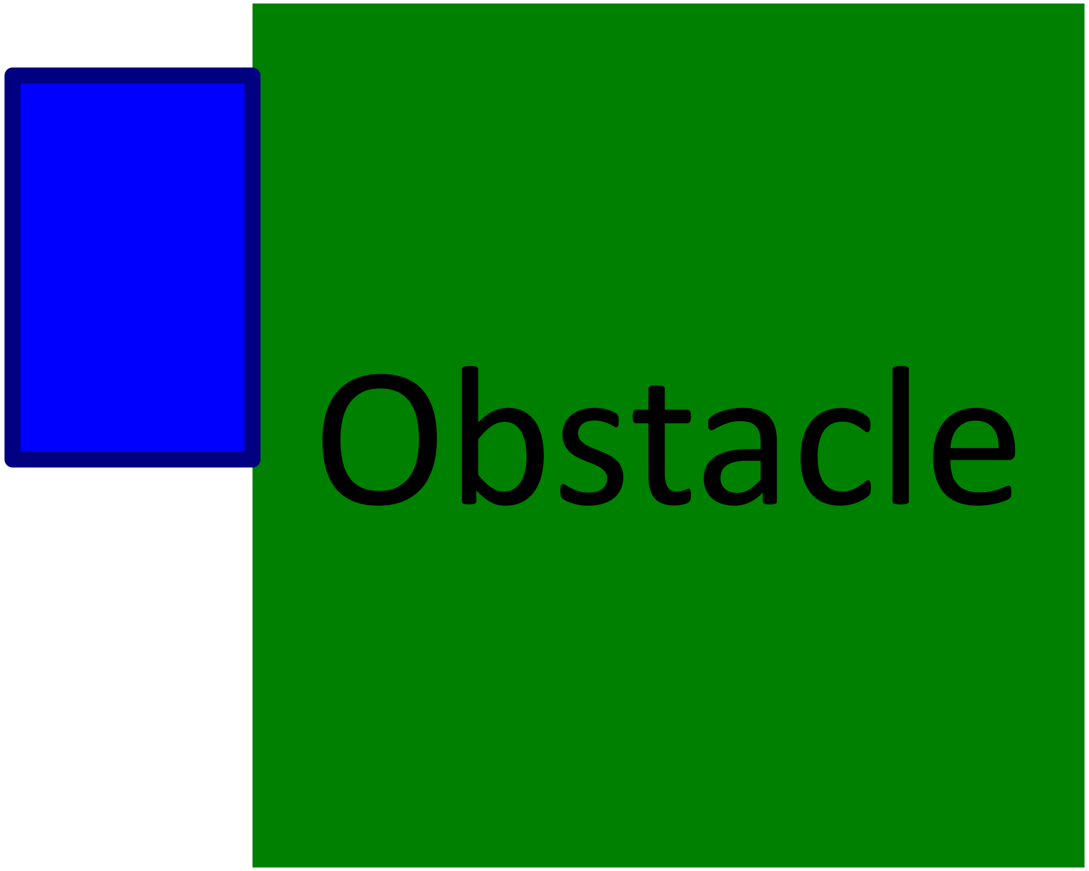

Obstacle

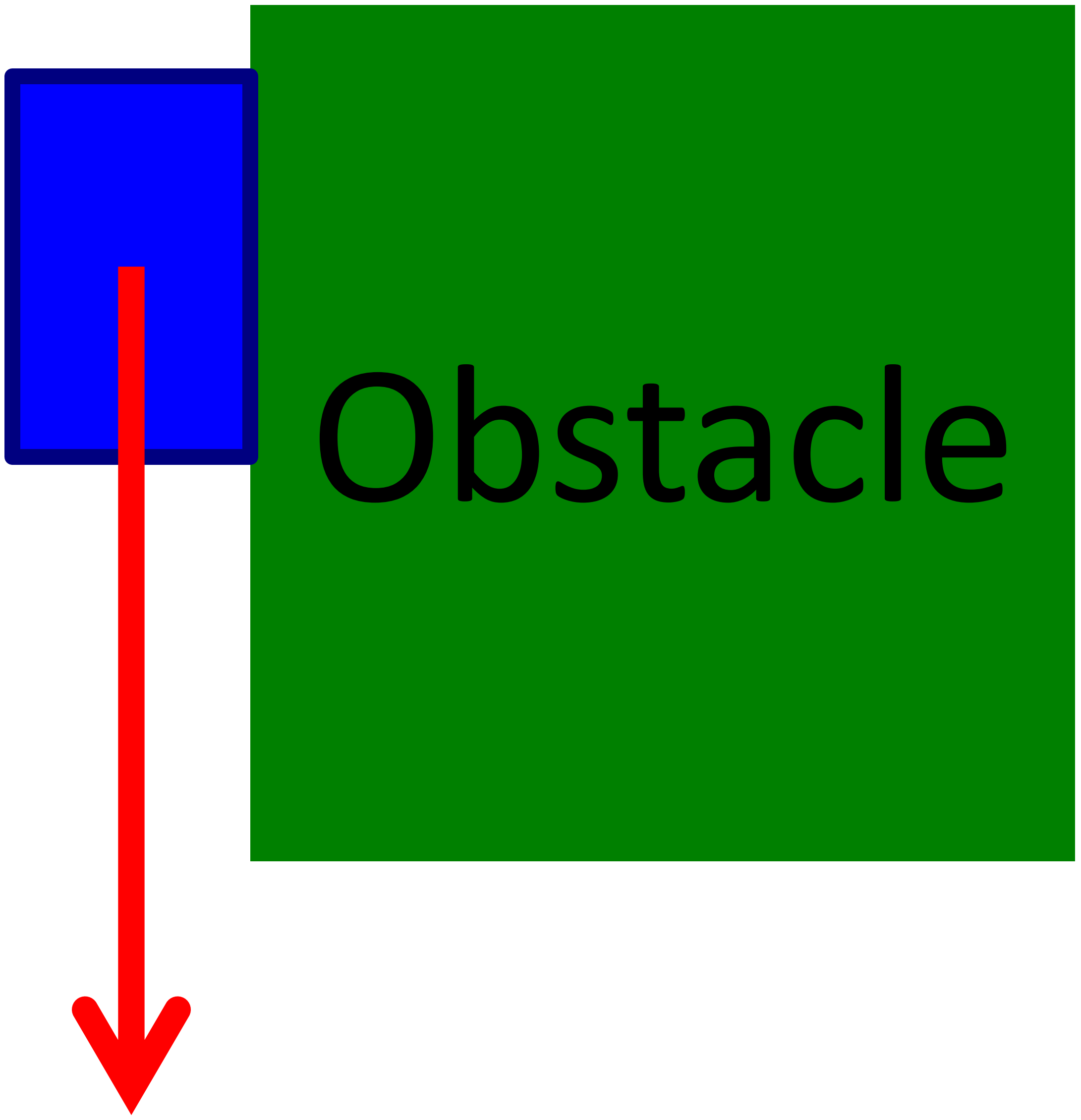

Obstacle

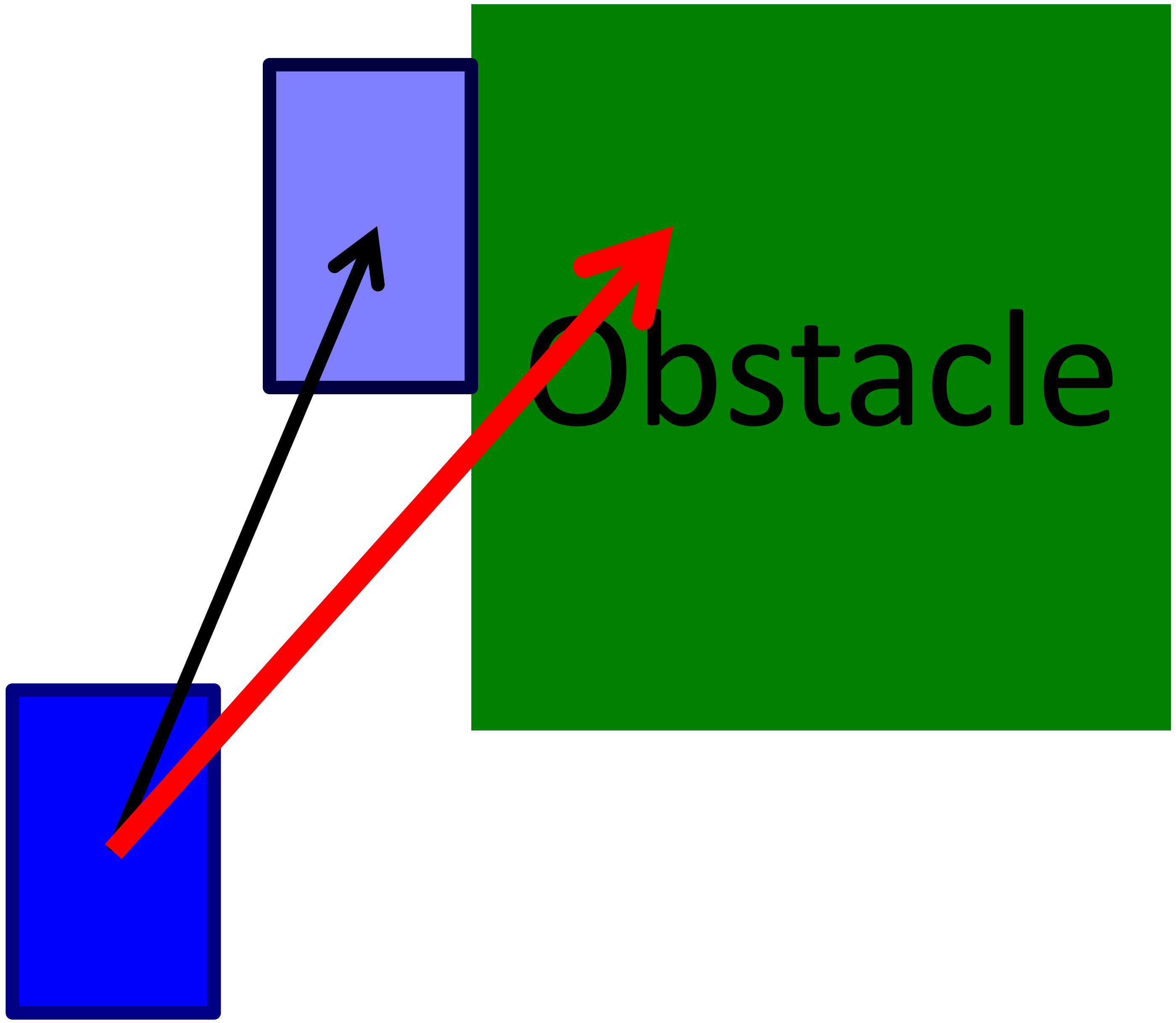

Obstacle

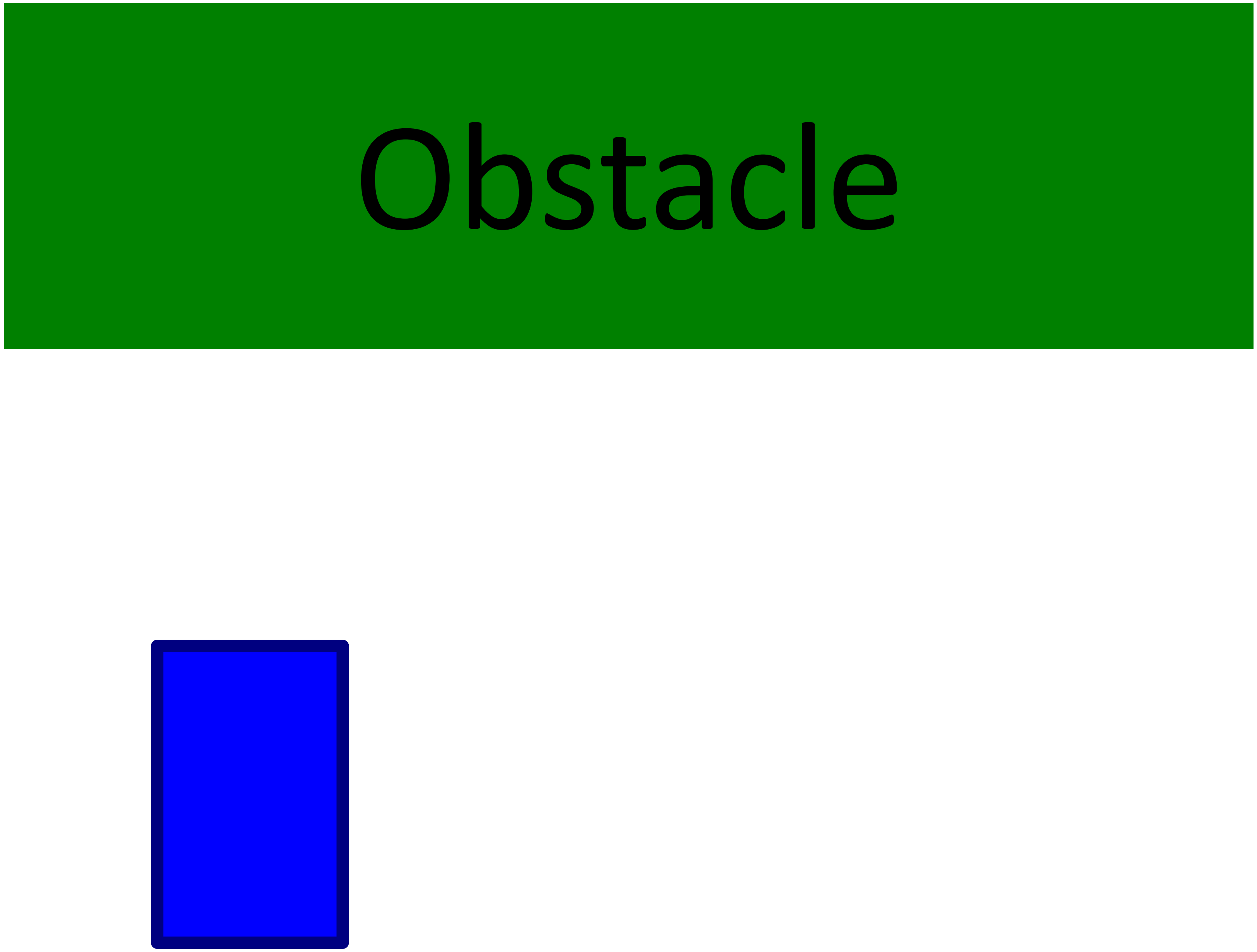

Obstacle

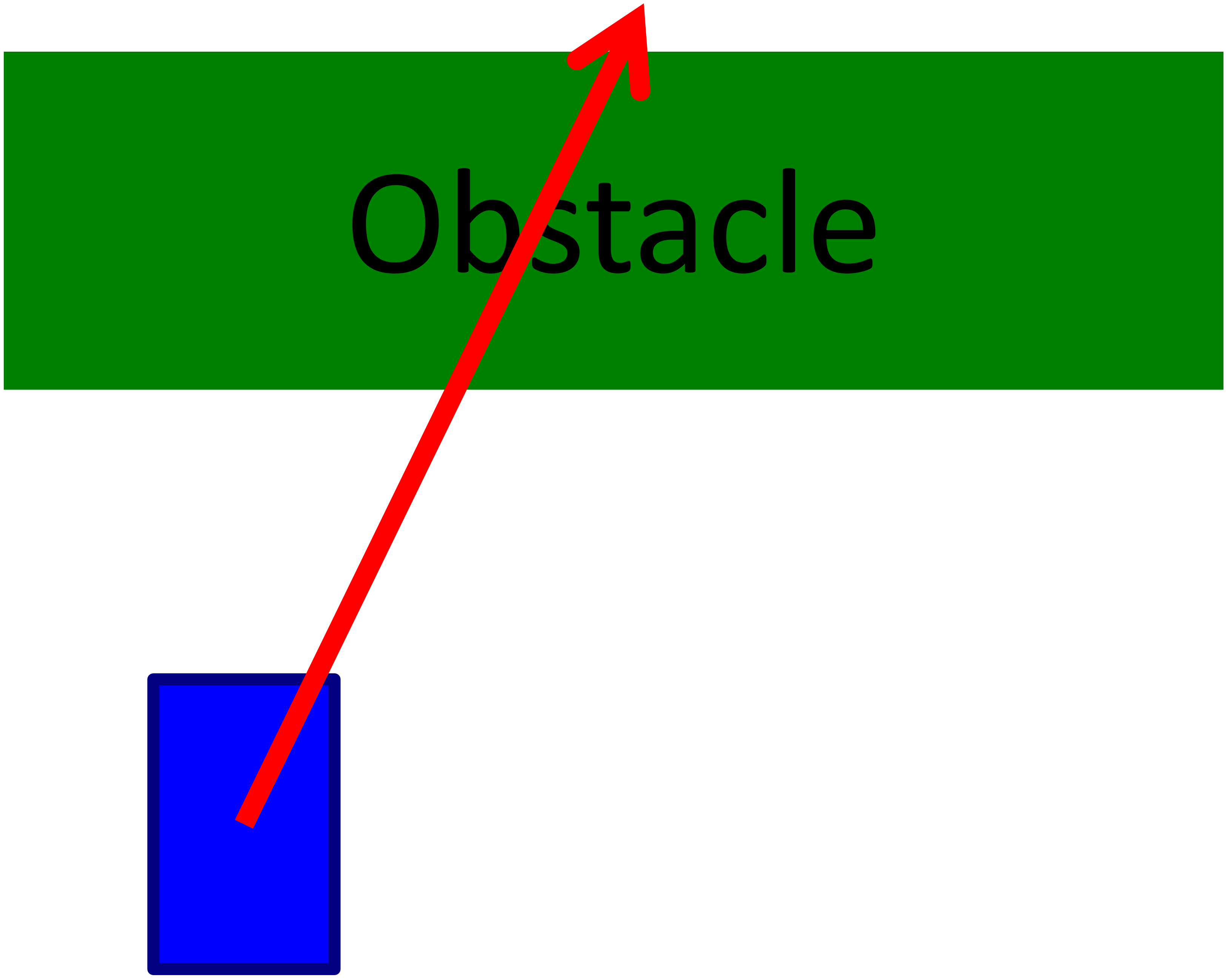

Obstacle

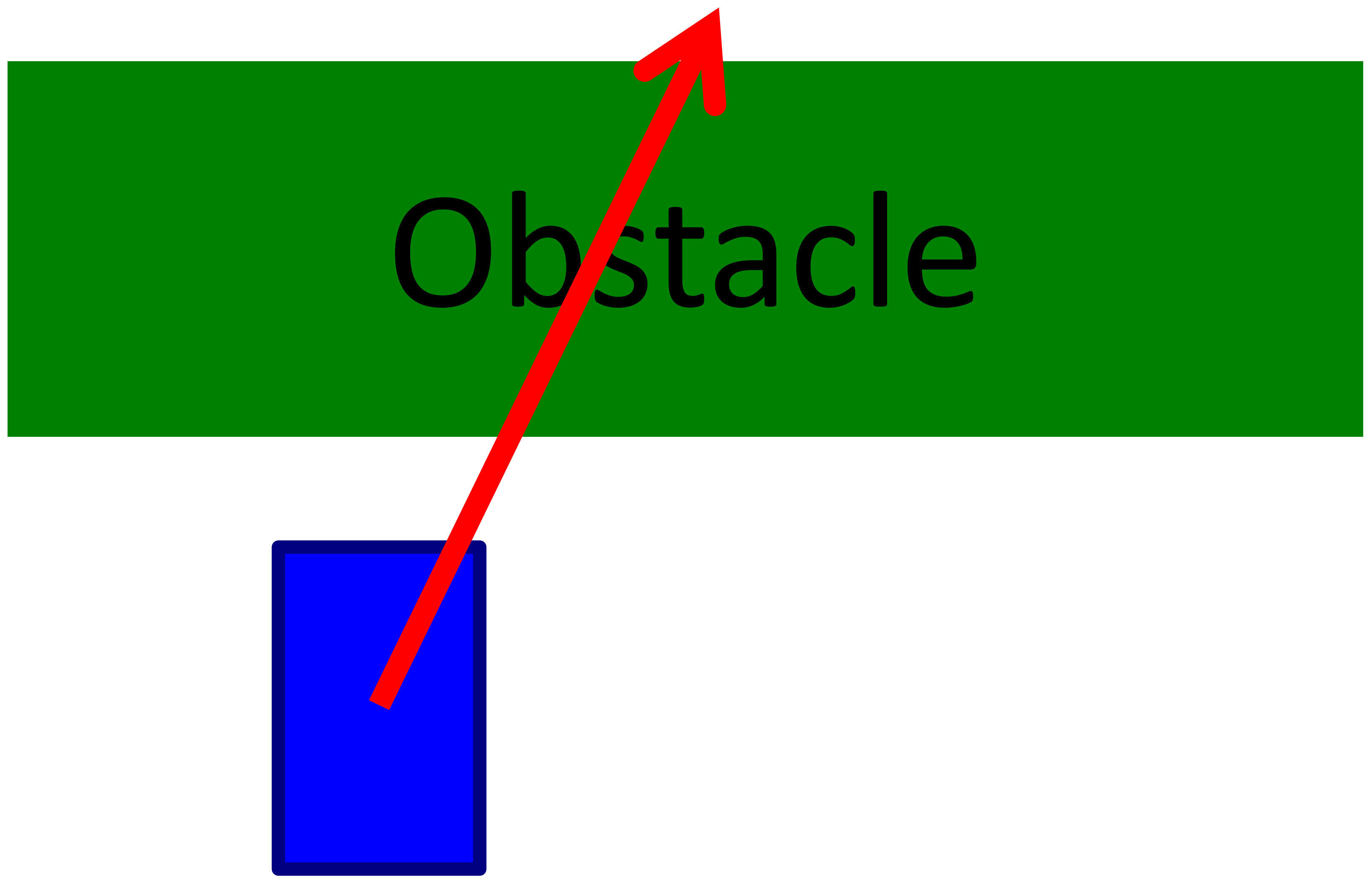

Obstacle

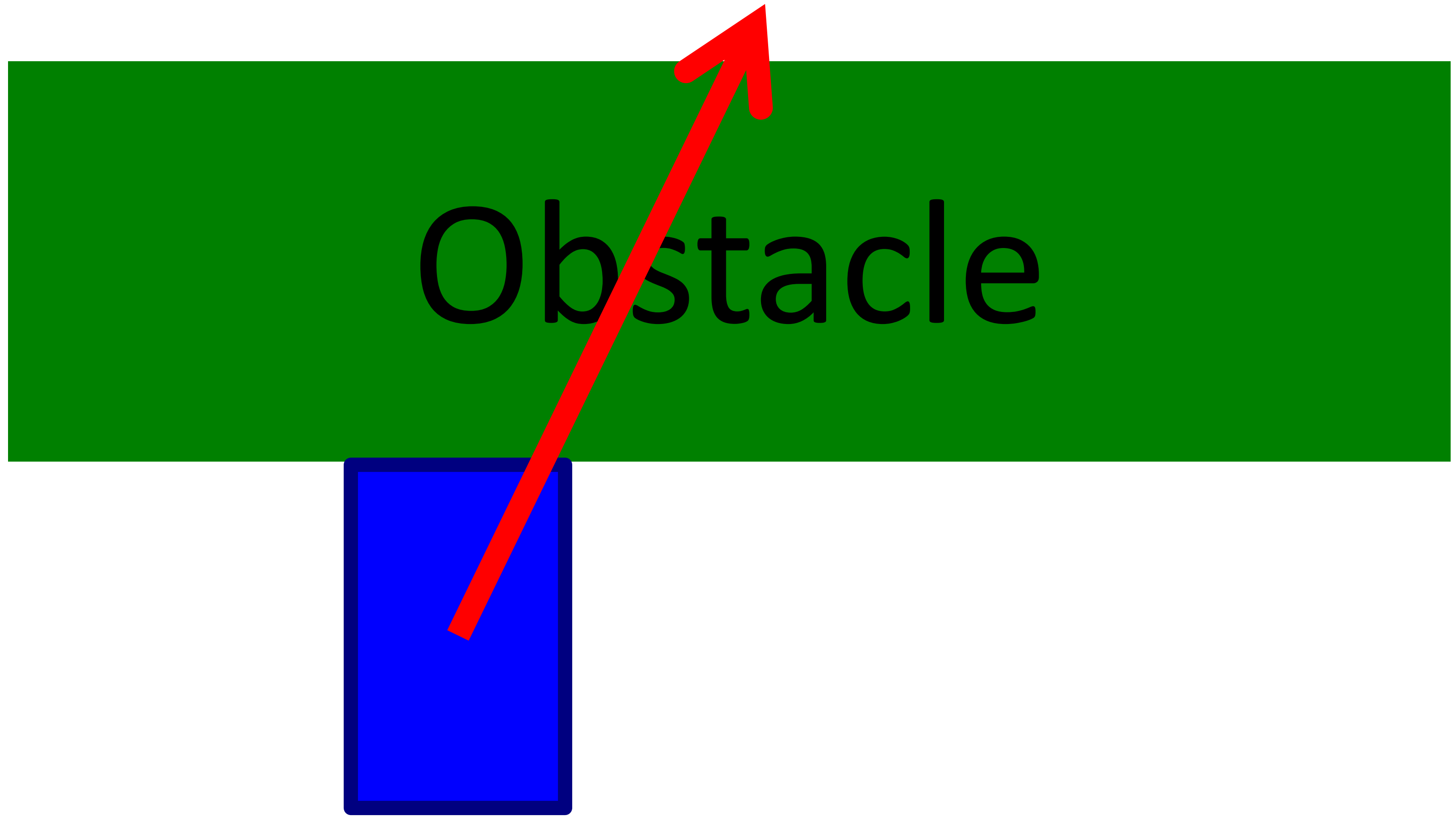

Obstacle

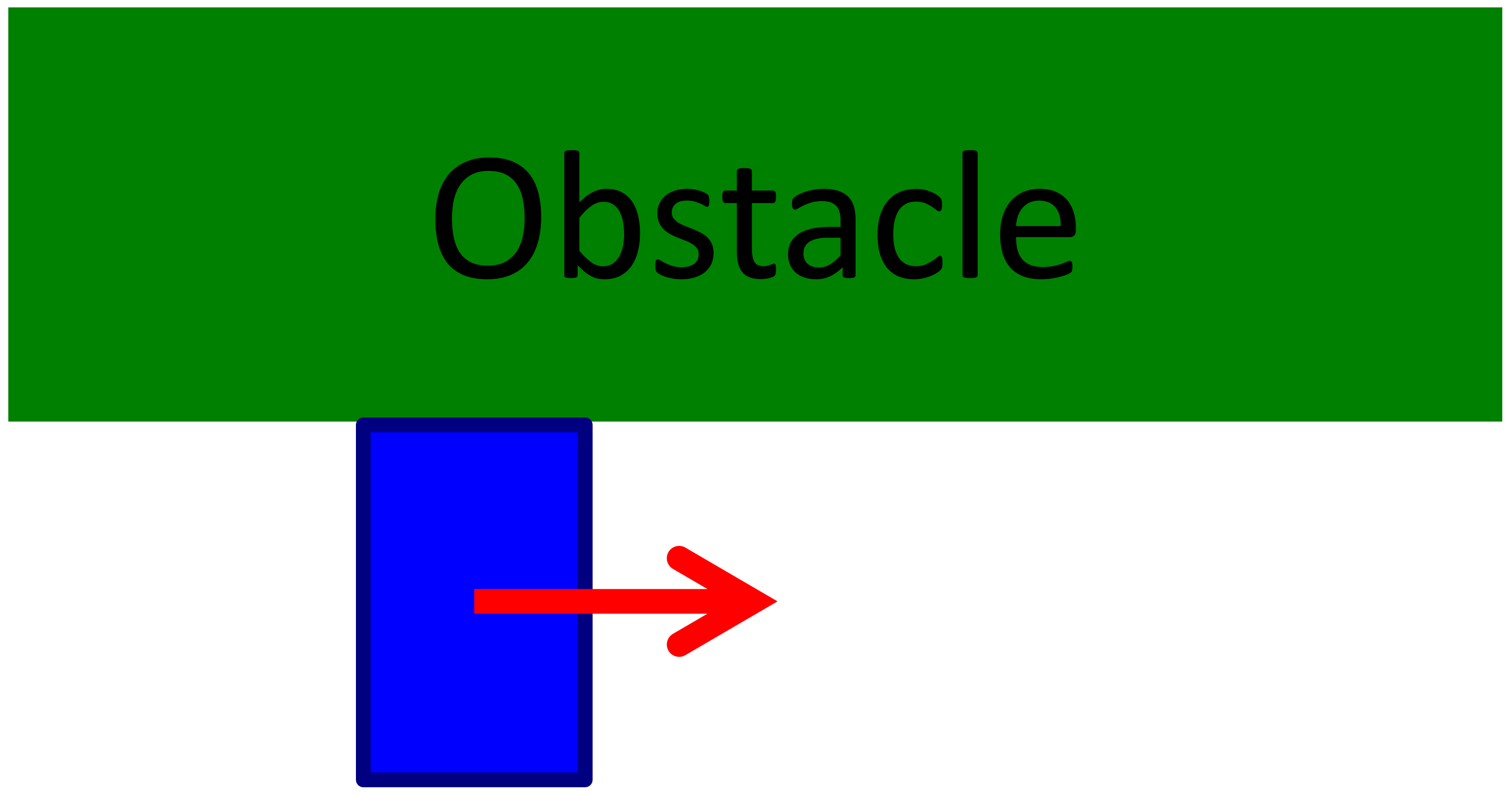

Obstacle

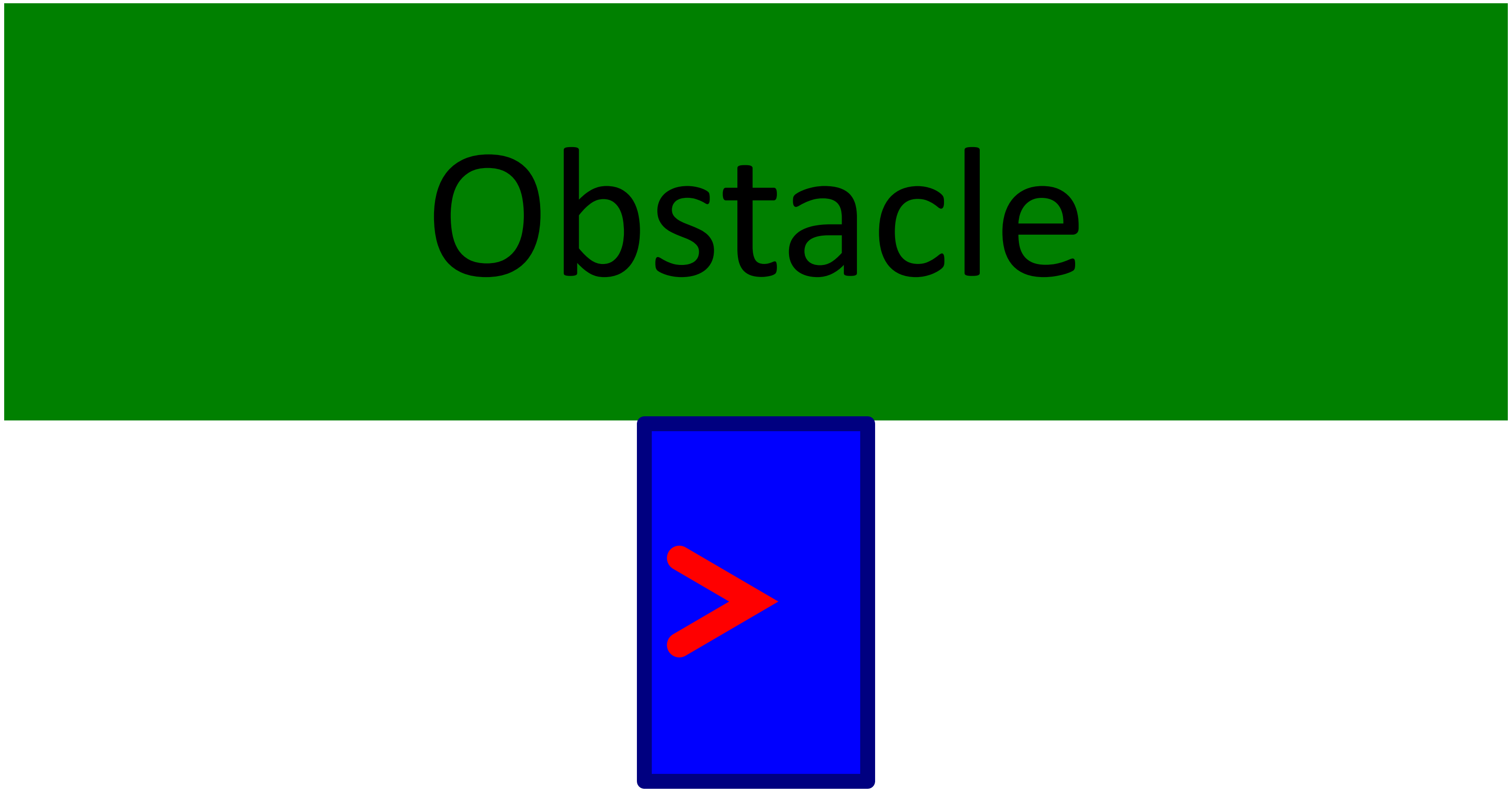

Obstacle

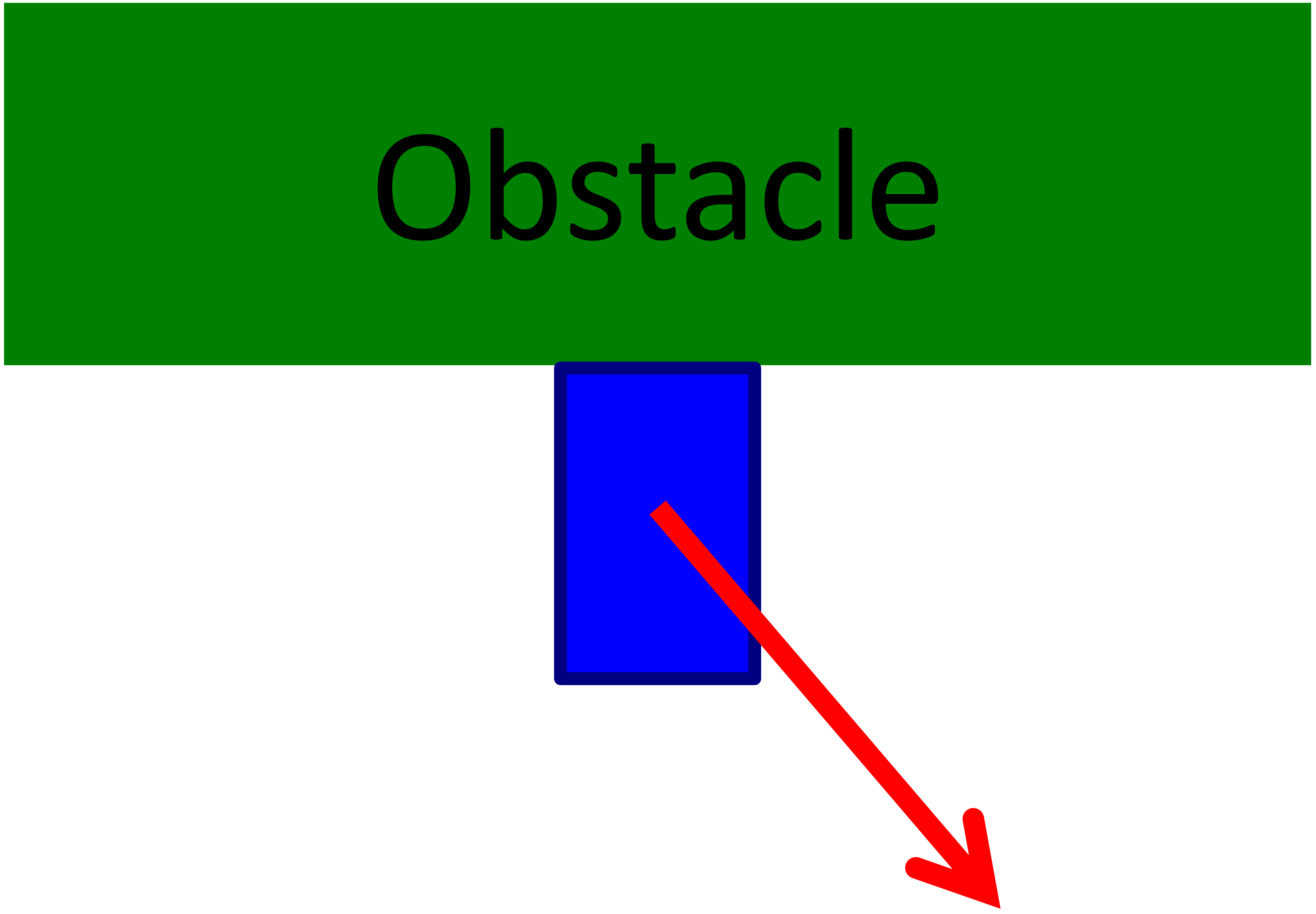

Obstacle

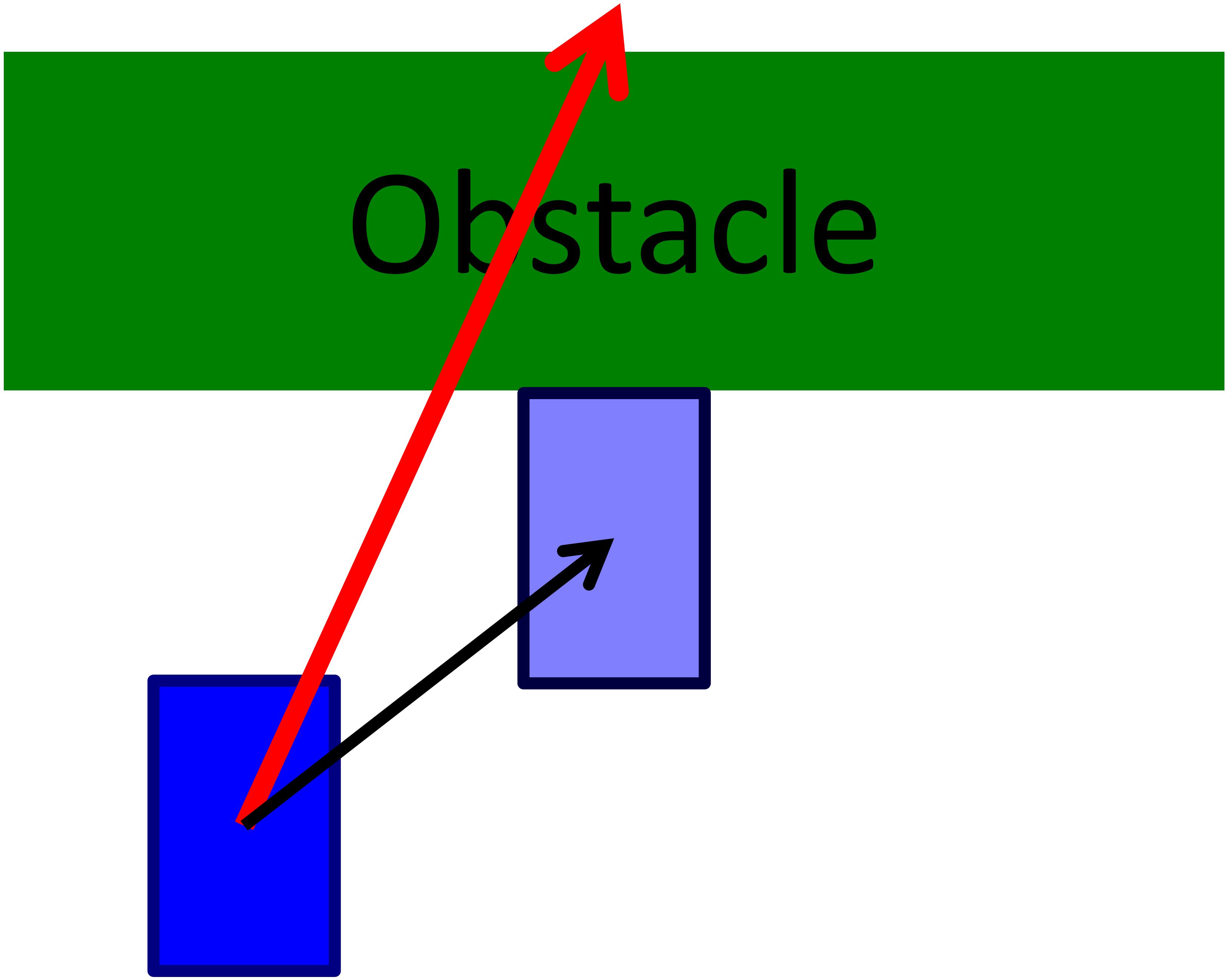

Obstacle


Obstacle

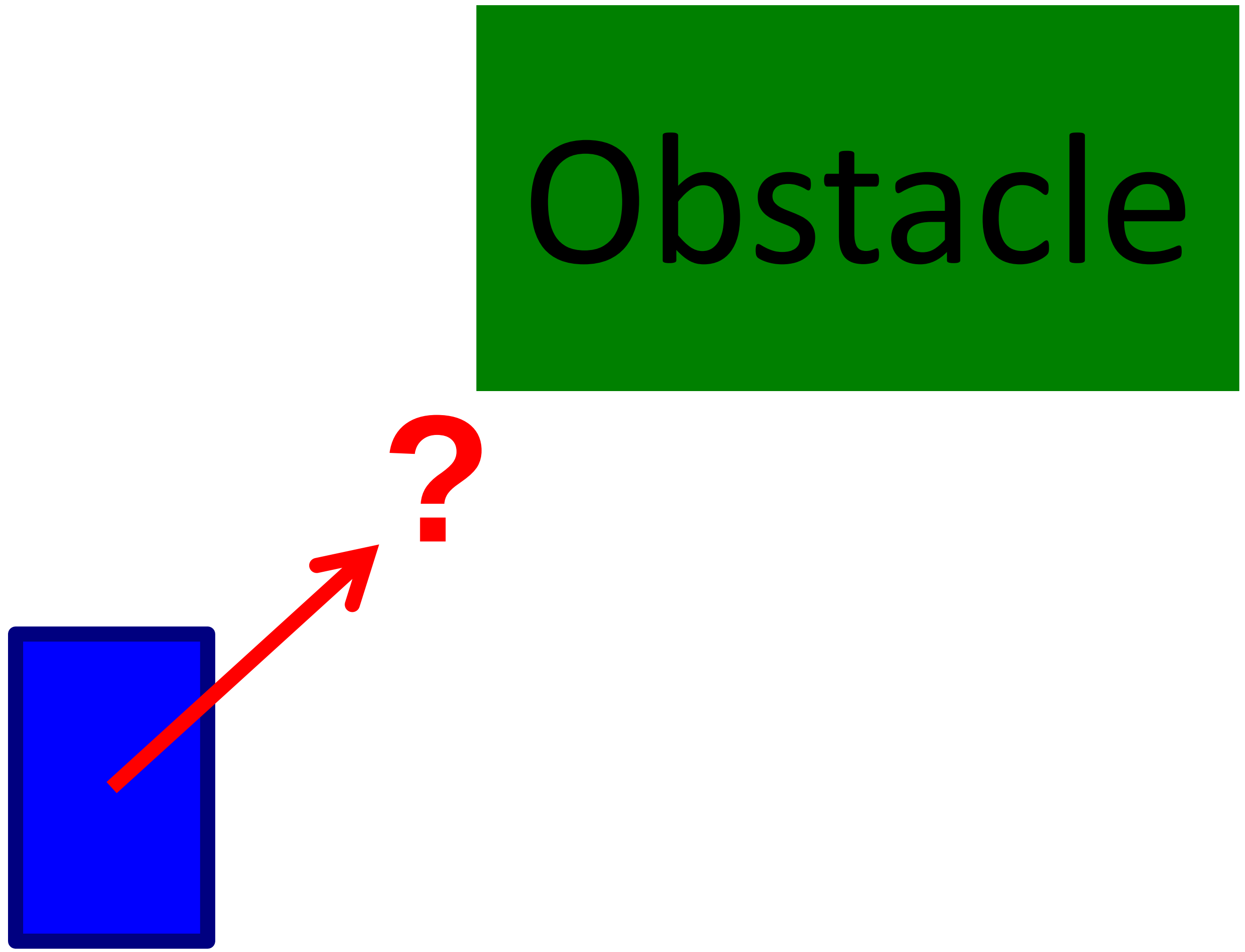

Obstacle

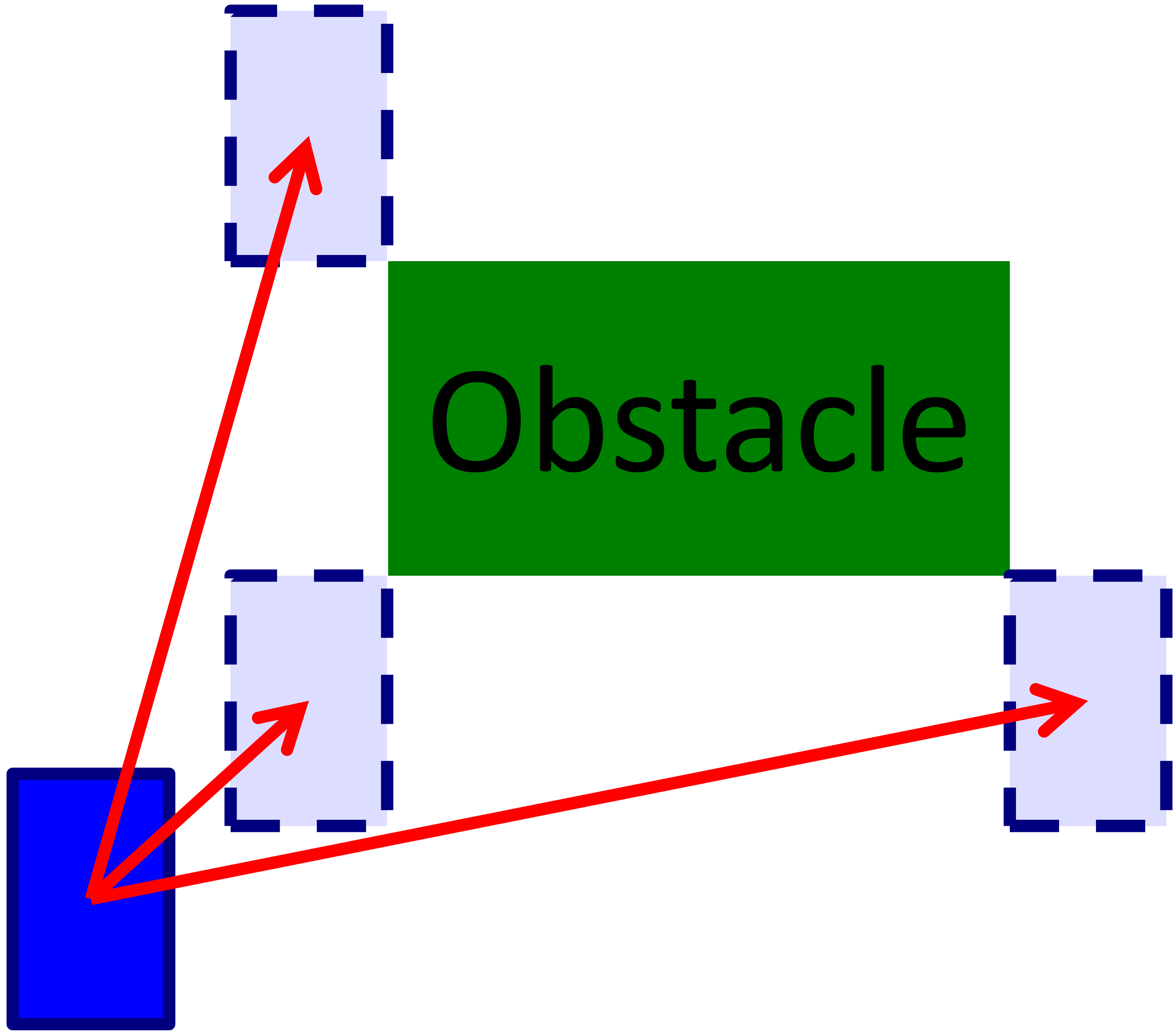

Obstacle

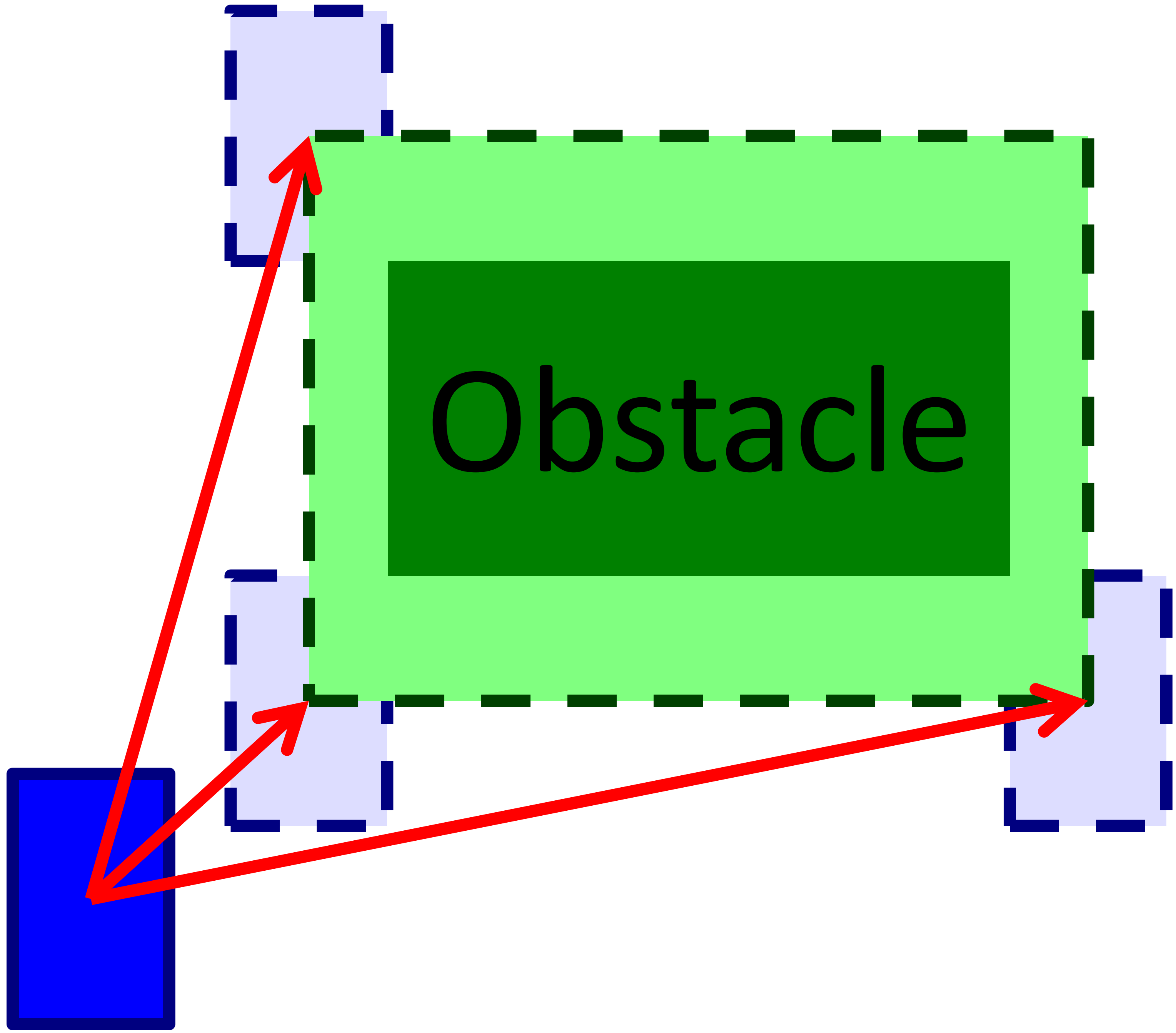

Obstacle

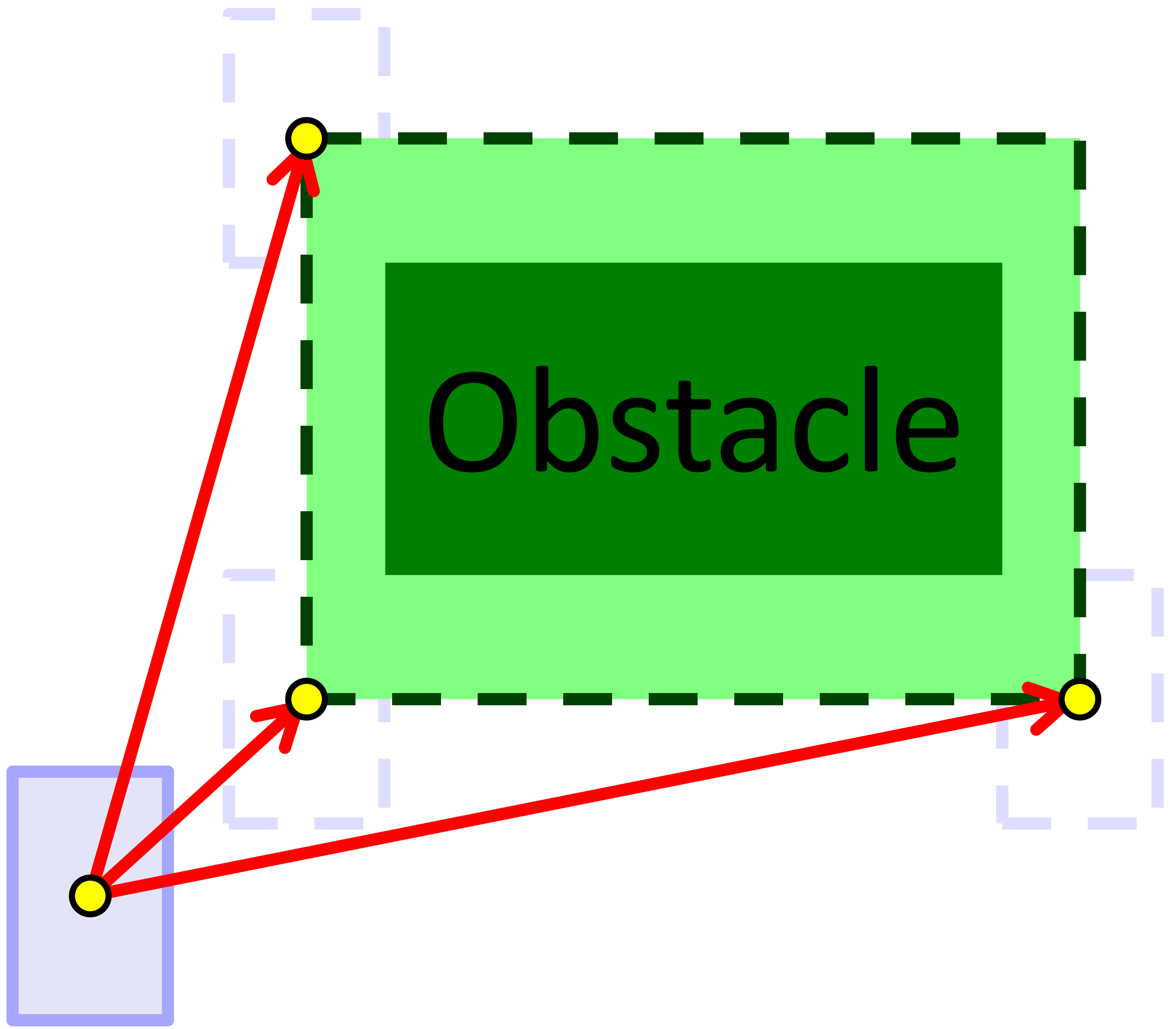

Obstacle

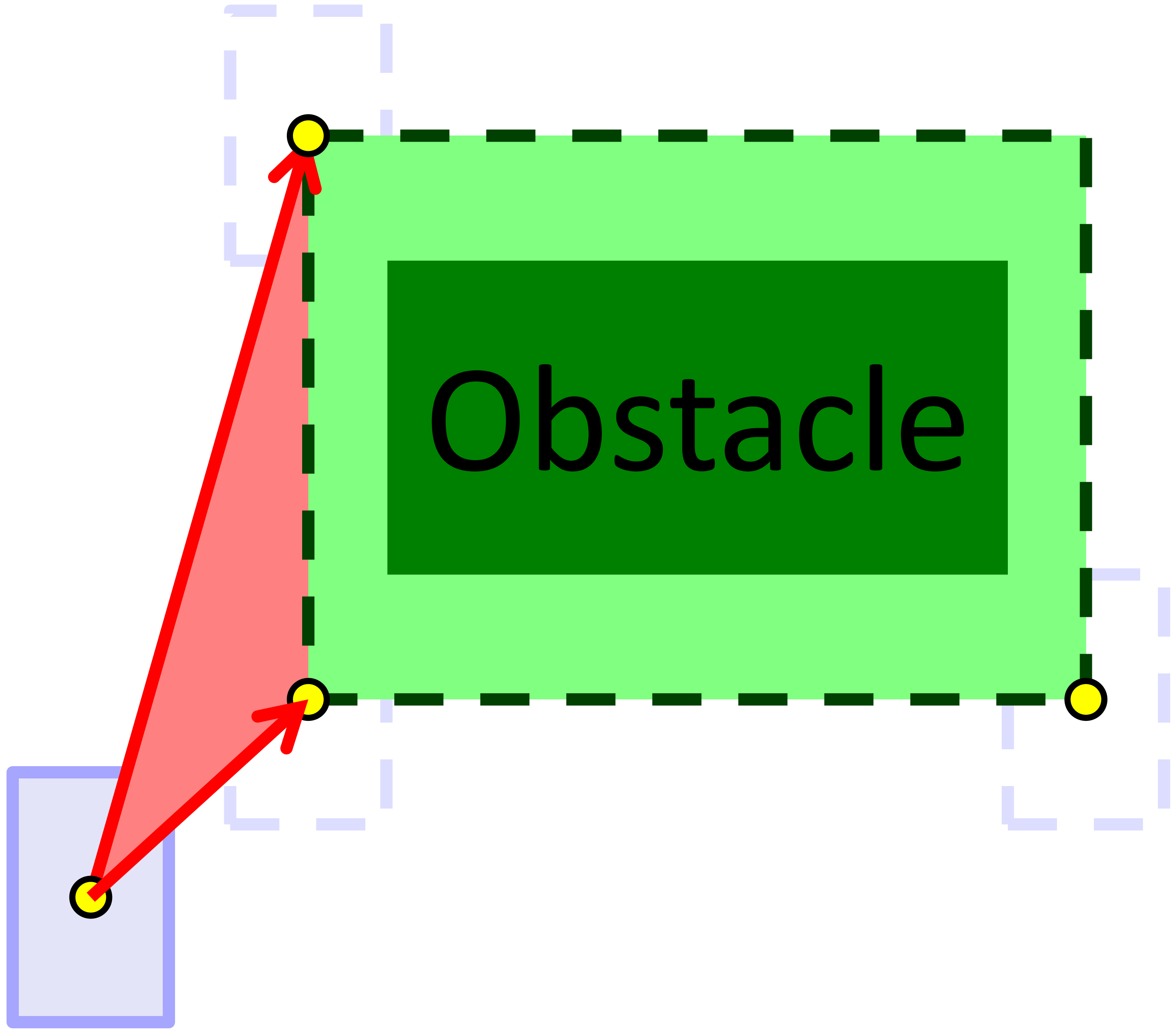
1. How do you detect collision?
2. What do you do upon collision?

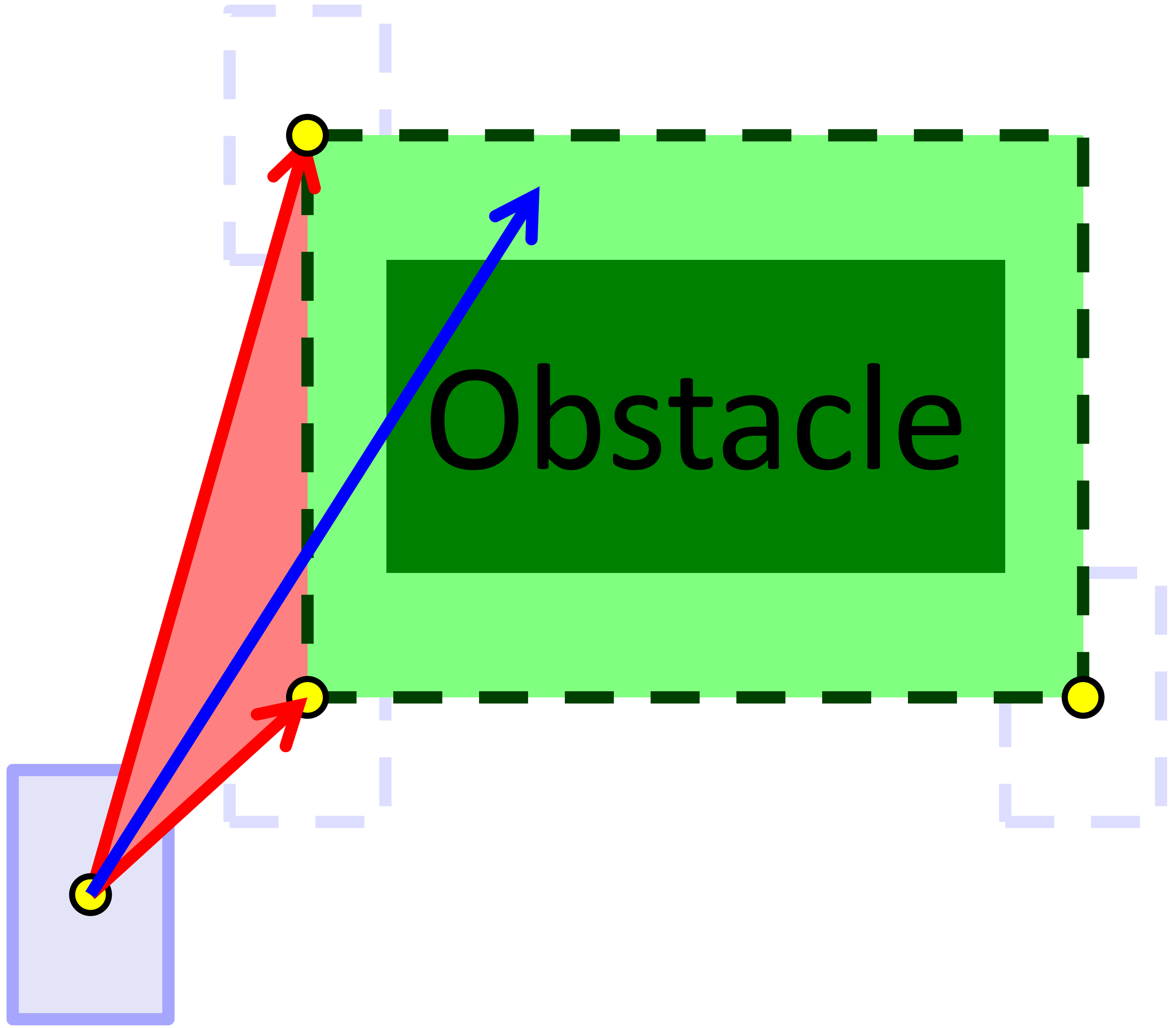

Obstacle
?

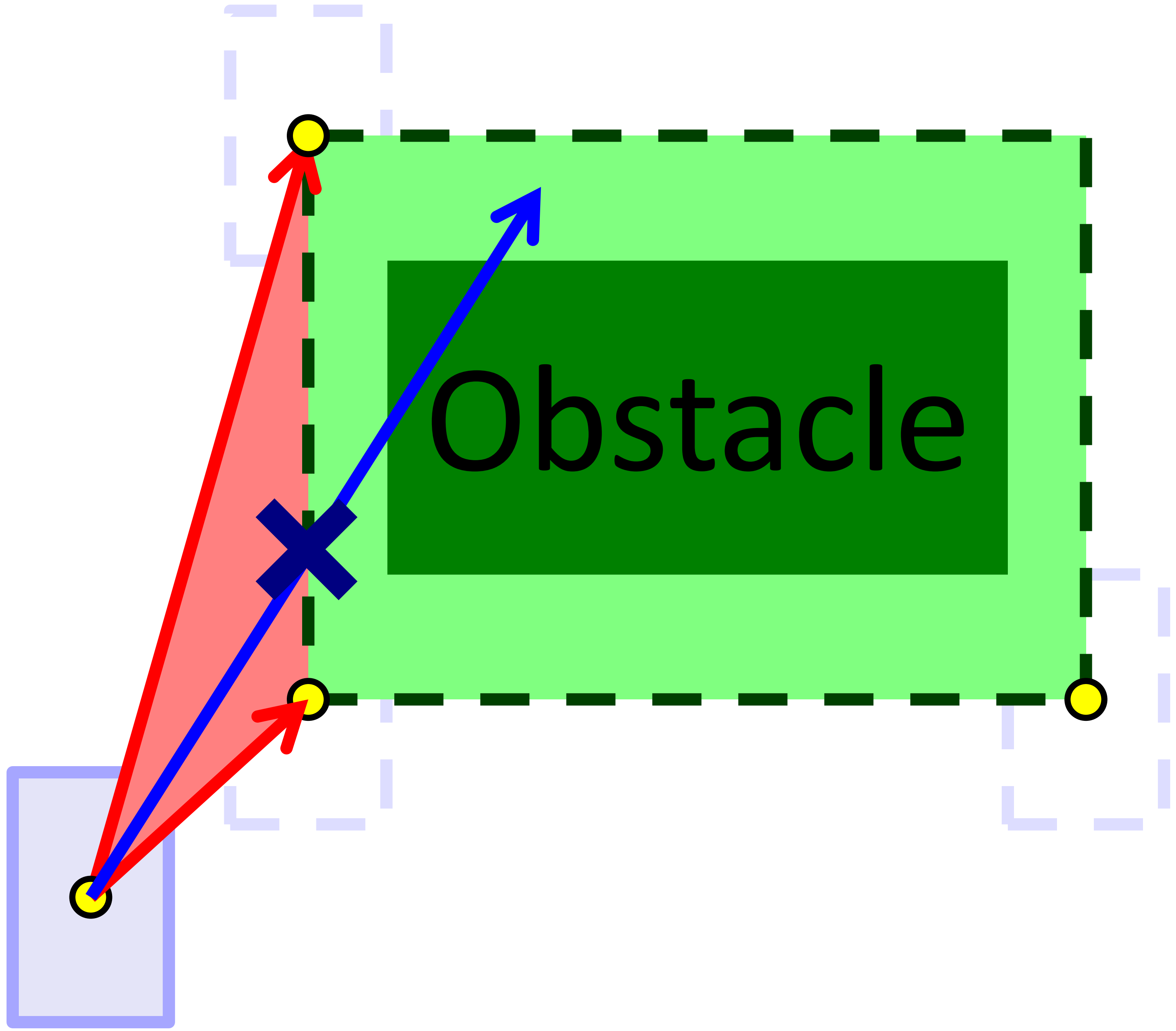

Obstacle

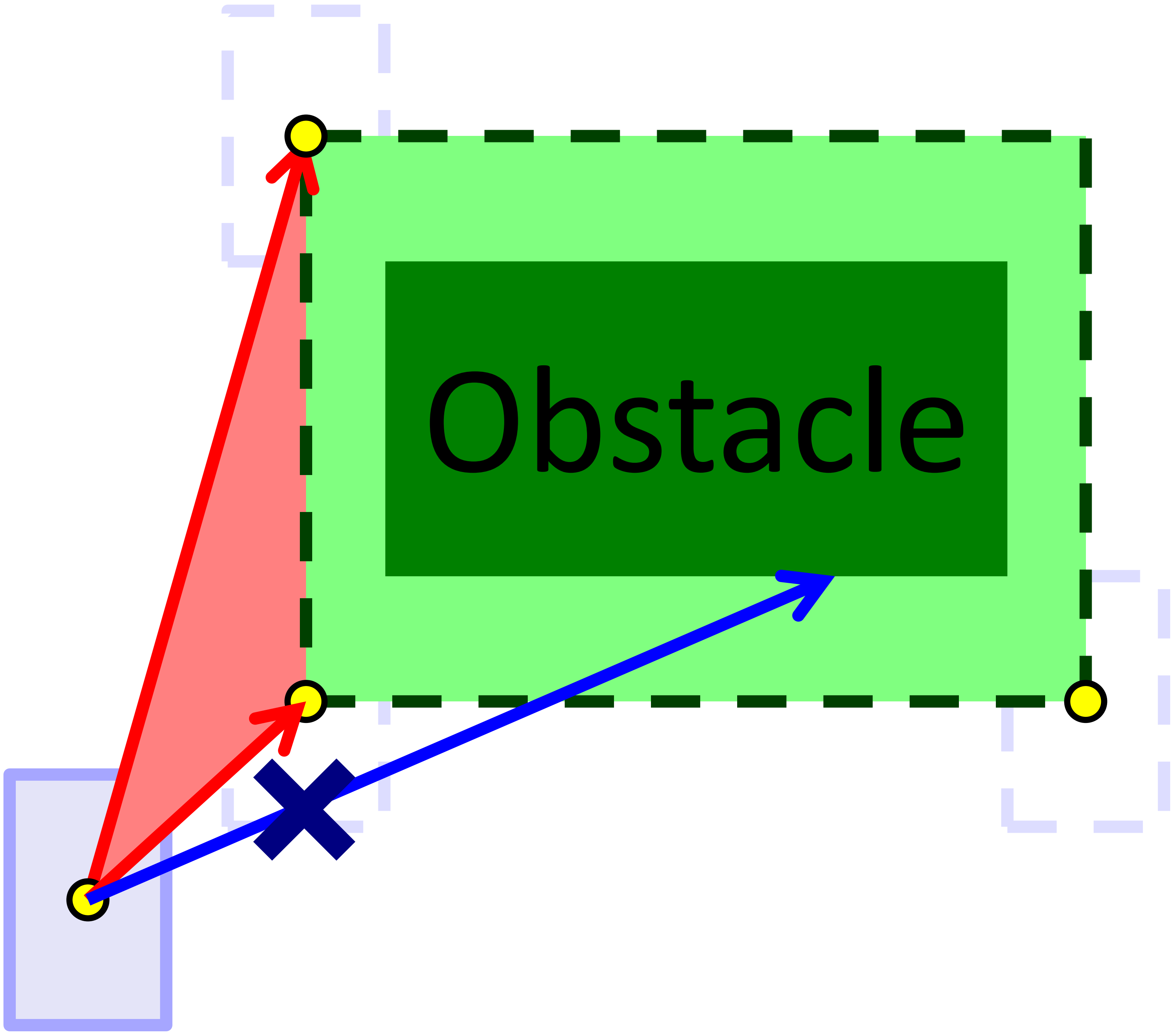

Obstacle

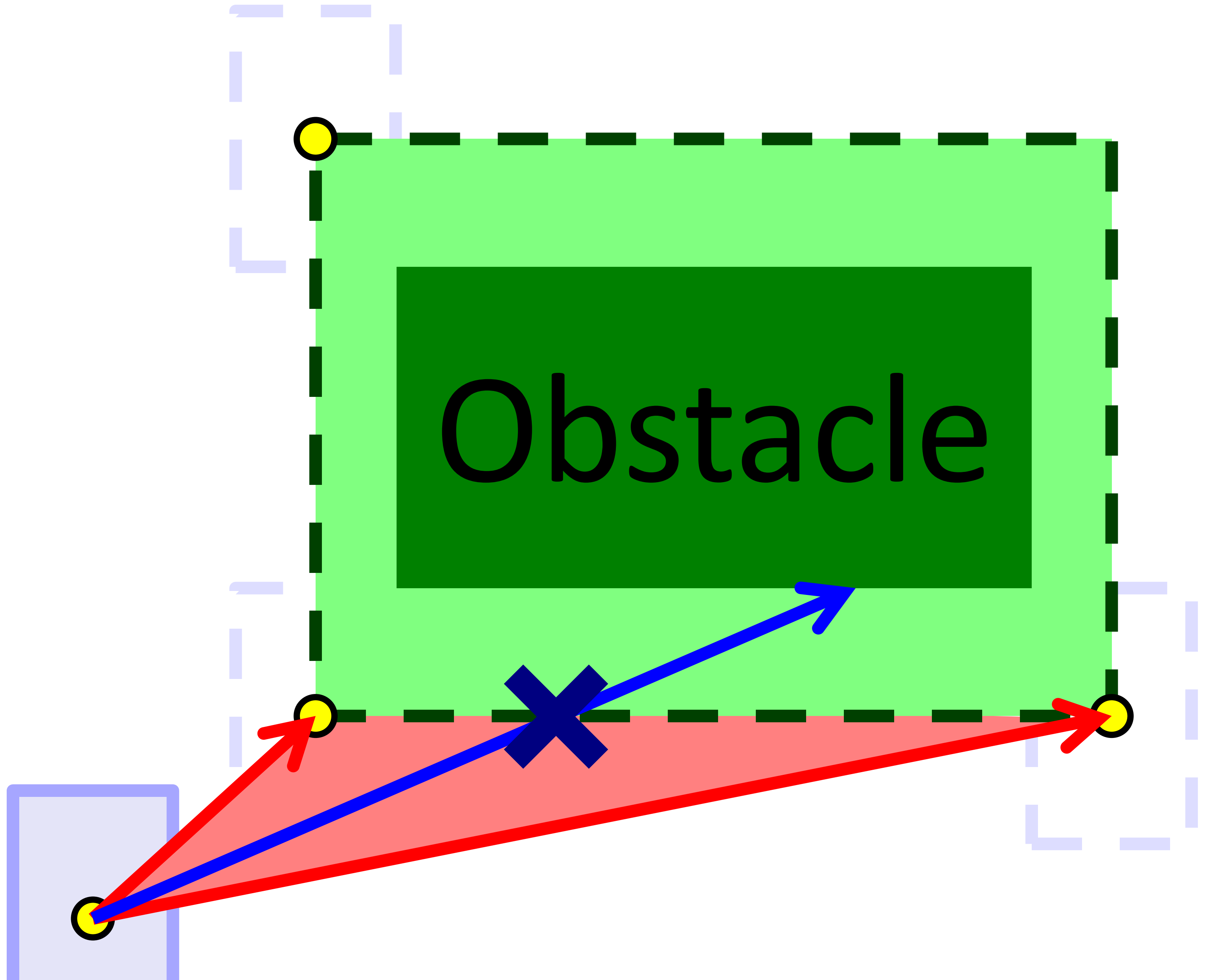

Obstacle


Obstacle


Obstacle


Obstacle


Obstacle


Obstacle

V9

Platforms

V10

Restart Button

End

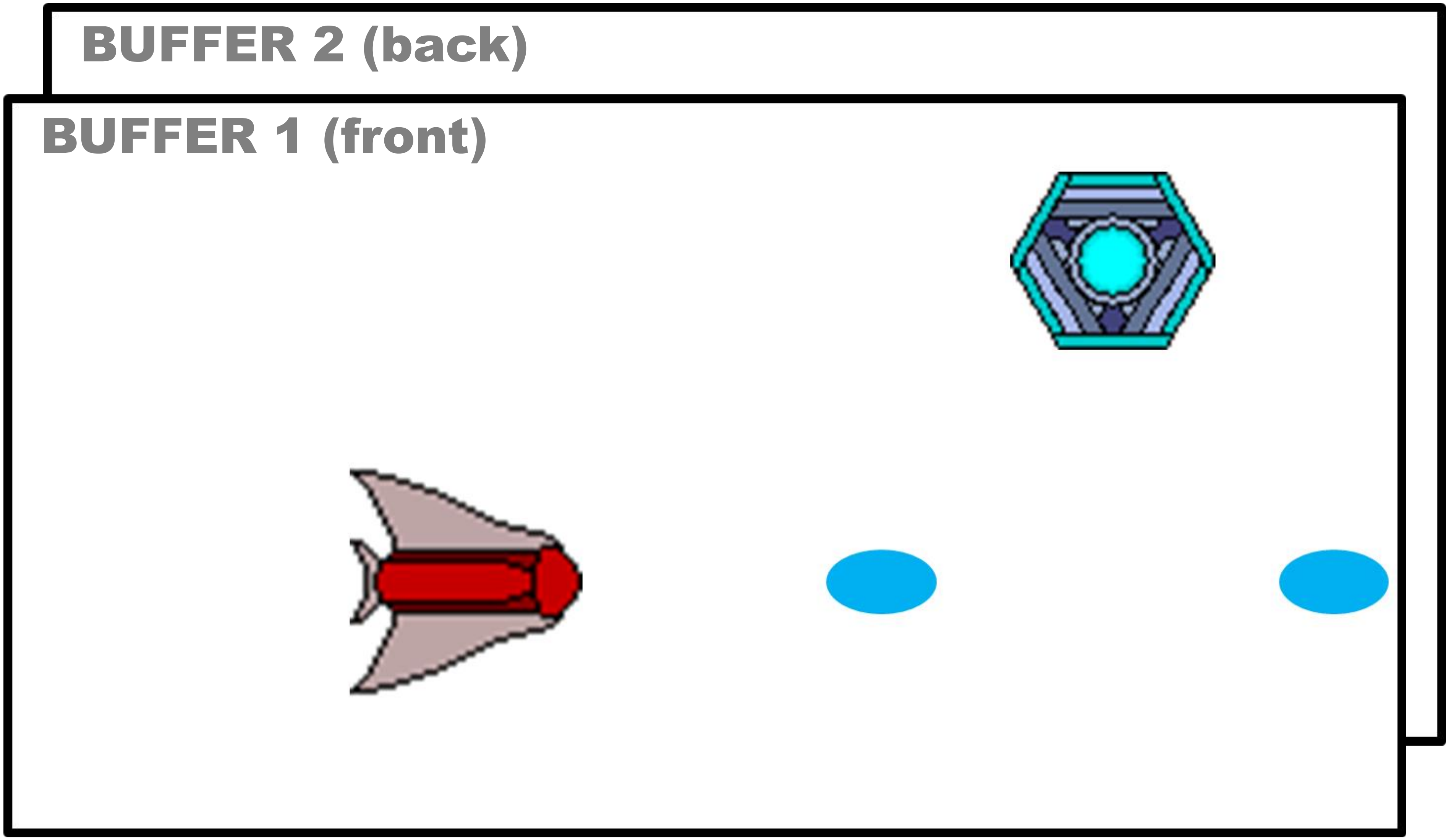
Similar Libraries / Frameworks

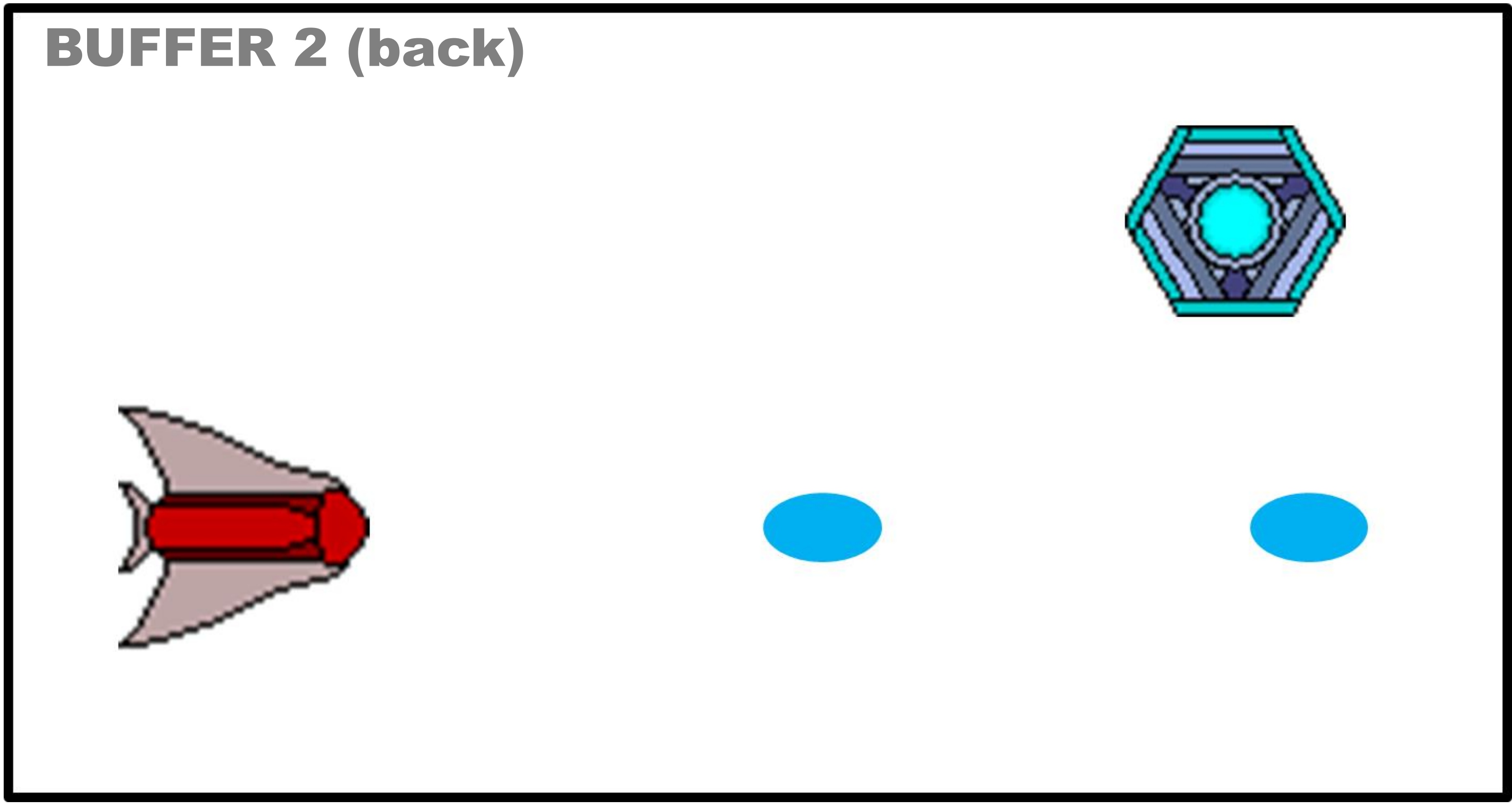
- Allegro 2D [C++]
- PyGame [Python]
- HTML5 canvas [javascript]

Some Notable Game Engines

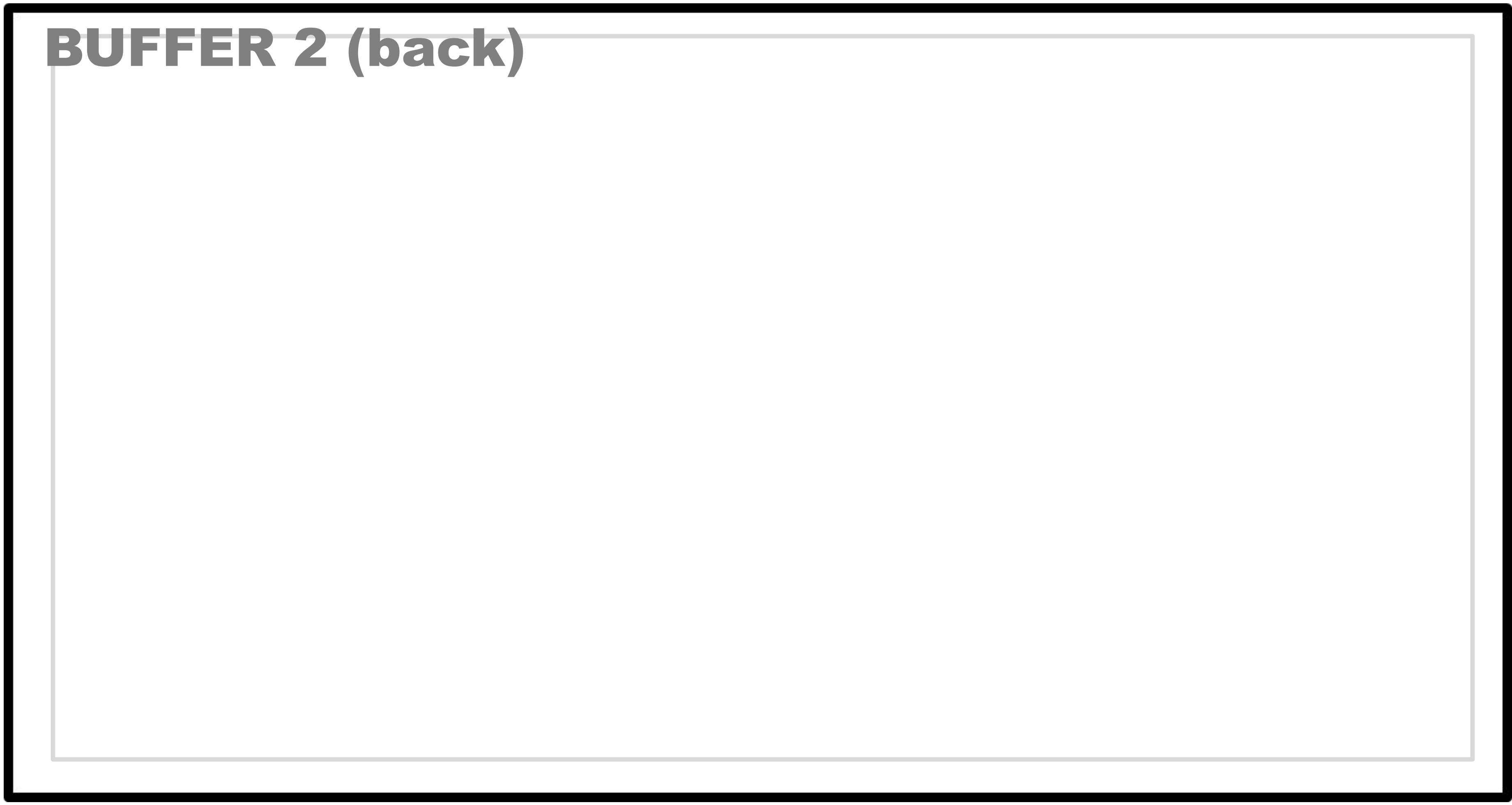
Unity	C#	cross platform
Monogame	C#	cross platform
libGDX	Java	cross platform
Cocos2d	C++/JS/Lua	cross platform
Construct 2	JS	mobile
Phaser.io	JS	web (HTML5)

Double-Buffering

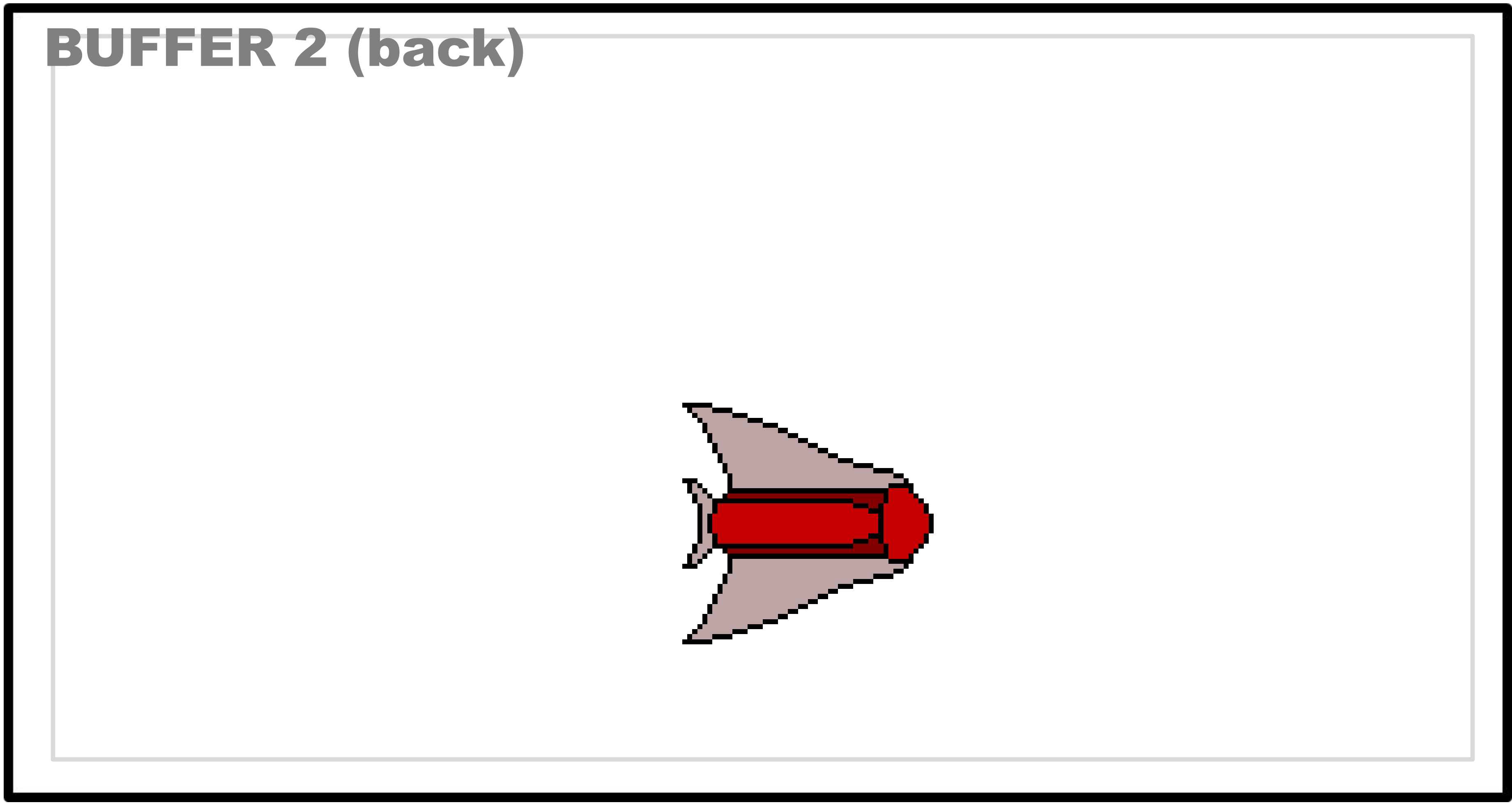

BUFFER 2 (back)
BUFFER 1 (front)



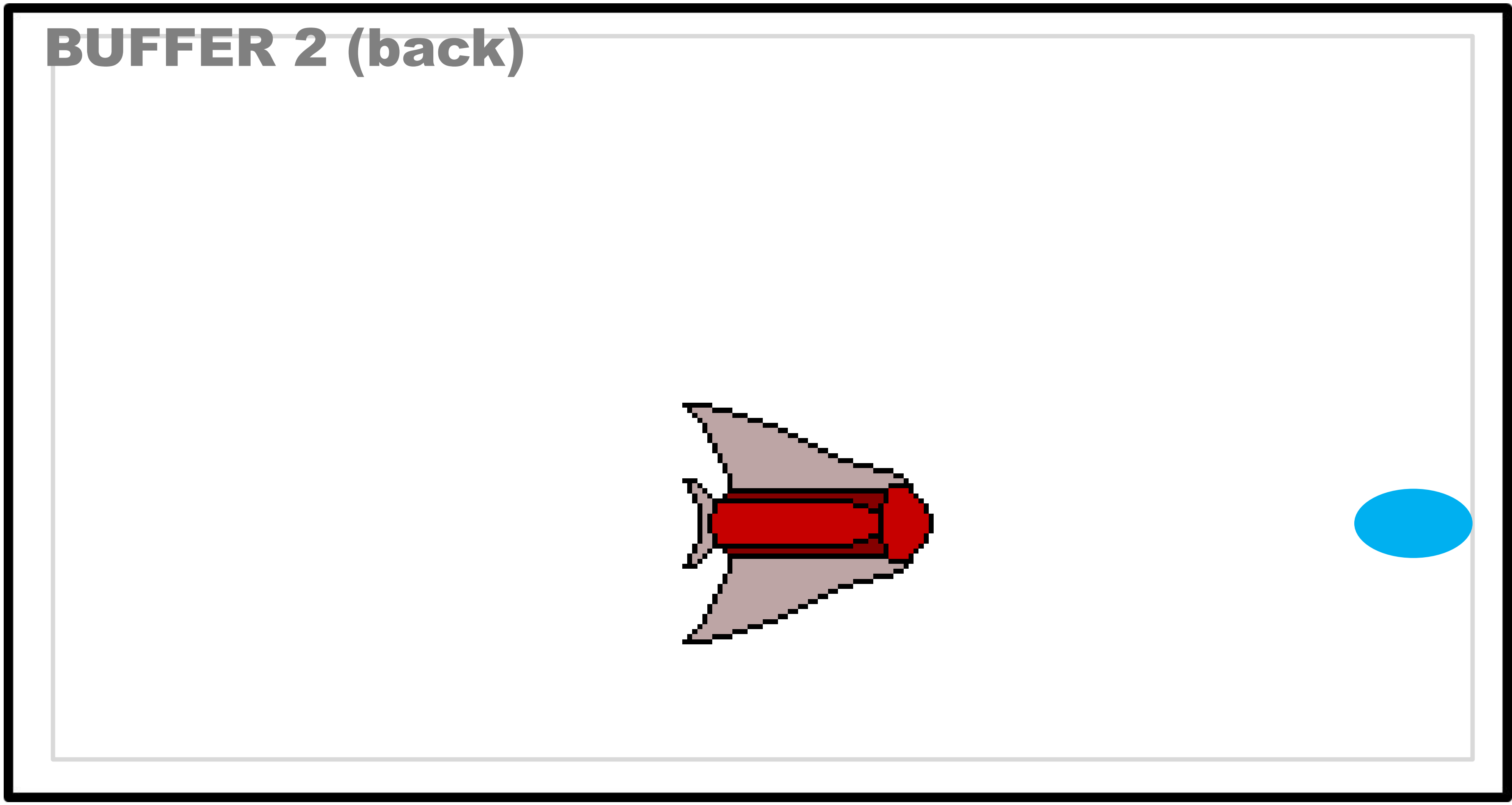
We work on the back buffer.
The front buffer is being displayed to the user.



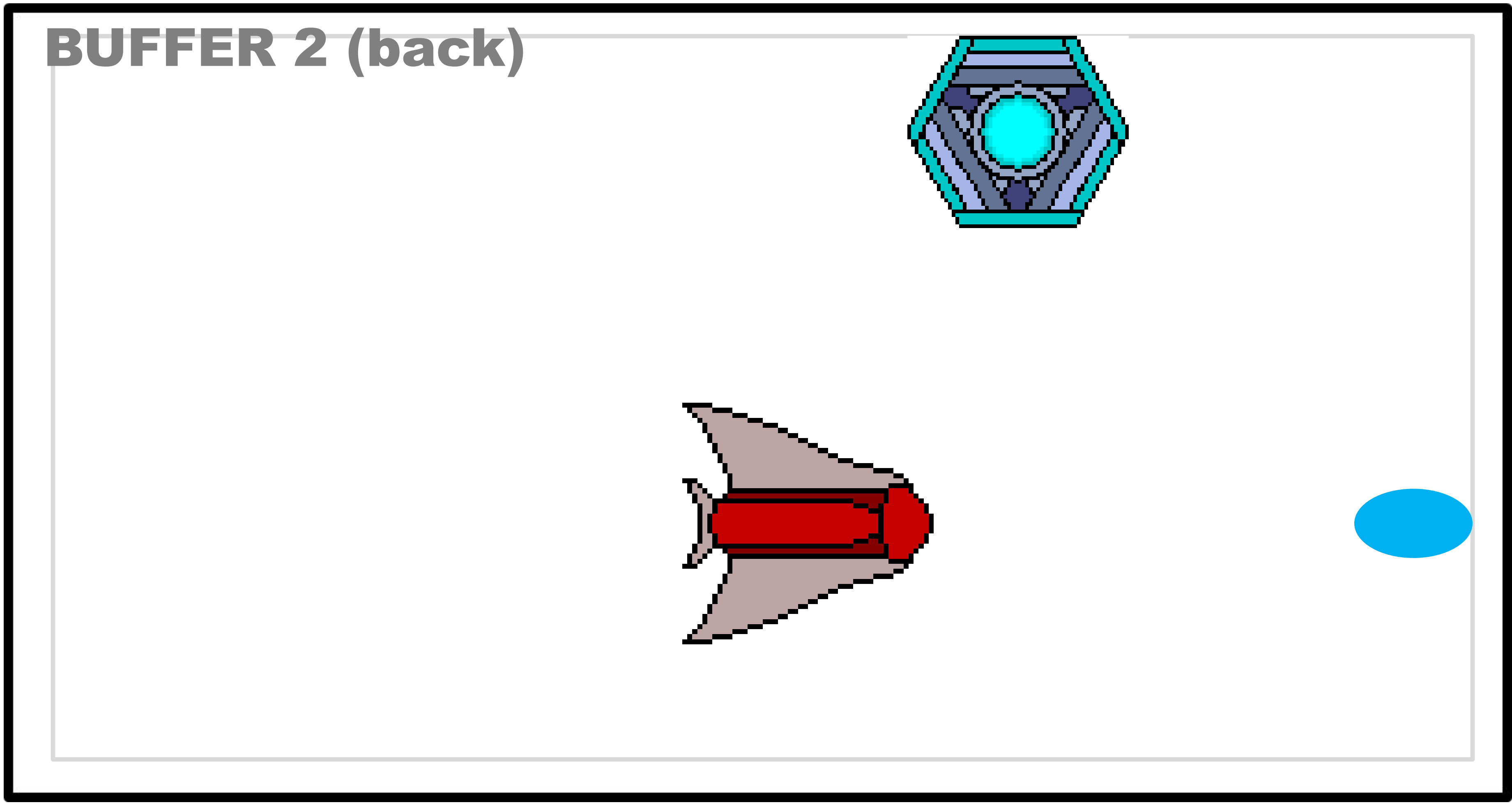
`window.clear()`



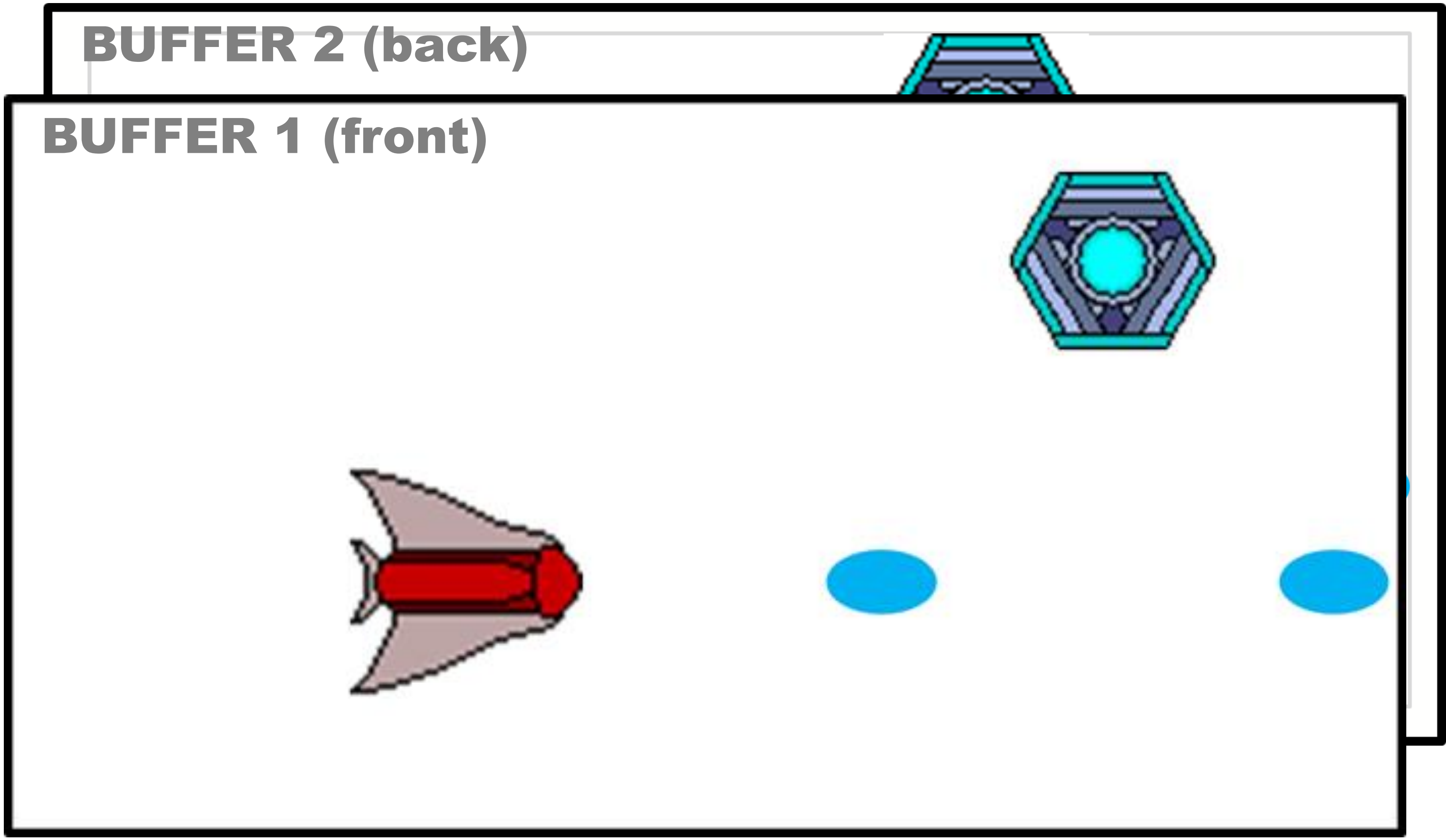
```
ship.draw()
```

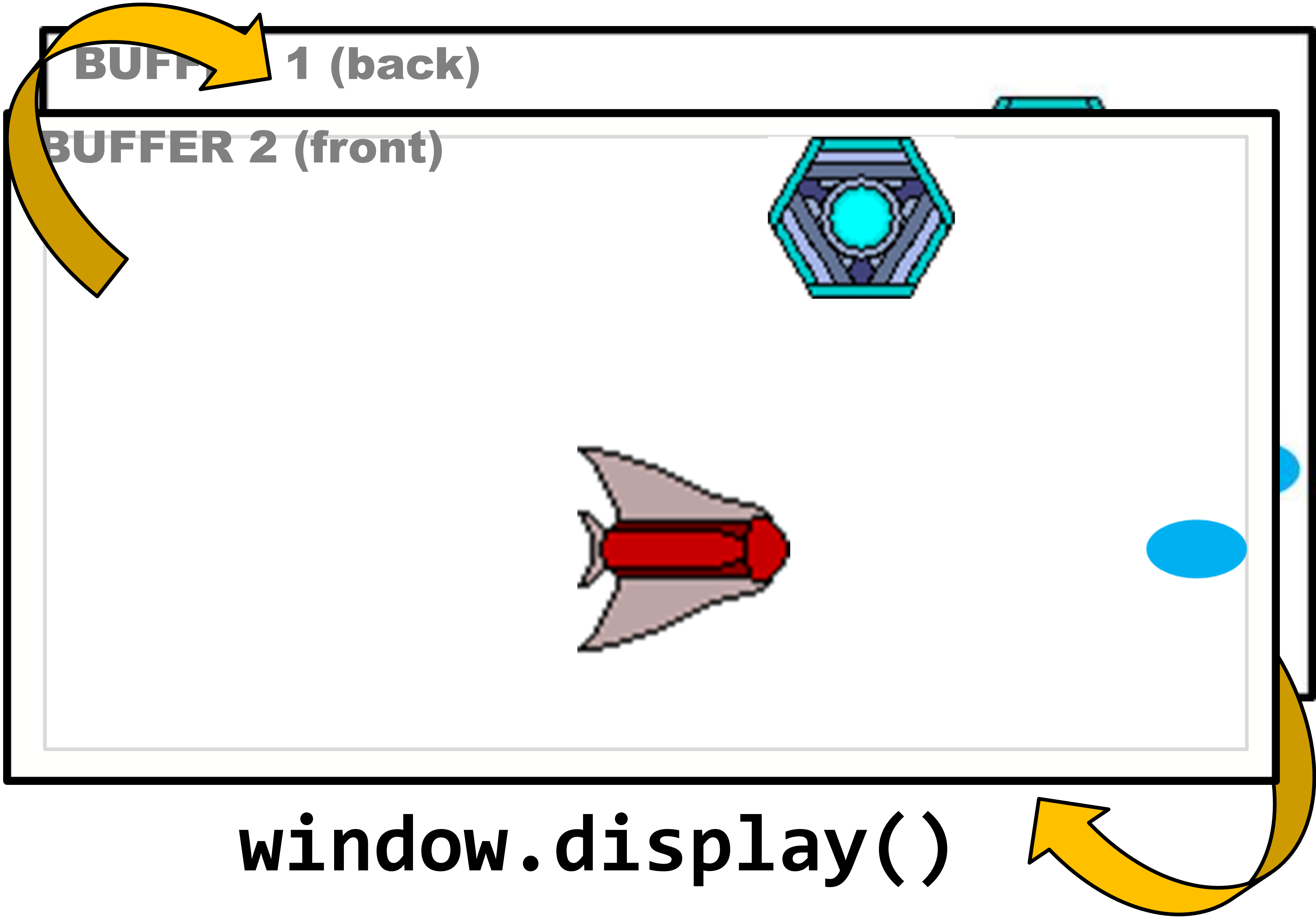


`bullet.draw()`



```
enemy.draw()
```


BUFFER 2 (back)
BUFFER 1 (front)



```
window.display()
```

ACKNOWLEDGEMENT

This presentation benefitted from

PowerPointLabs

a PowerPoint plugin for creating
better presentations with less effort.

PowerPointLabs
is available for free at
http://PowerPointLabs.info