



Game Design 3: Game Math and Collisions

Coordinates, velocity, delta time, gravity and bounding boxes

NAME _____

DATE _____

CLASS _____

Coordinates, Velocity and Time

Many 2D engines measure **screen coordinates** in pixels with the **origin (0, 0) at the top-left corner**. **x increases to the right** and **y increases DOWN** the screen, so moving *up* means y gets smaller. (Math class, and some engines, put y up. Always check which one your engine uses.)

A game redraws the screen many times per second. Each redraw is a **frame**, and the **frame rate** is how many frames per second (**fps**) the game draws. Every frame, the **update loop** moves each object by its **velocity**: $x = x + vx$ and $y = y + vy$.

If velocity is measured in pixels *per frame*, the game slows down on a computer that draws fewer frames. To prevent this, engines provide **delta time** (**dt**): the number of seconds since the last frame. Store speed in pixels per **second** and move by **distance = speed x dt** each frame. At 60 fps, **dt** = 1/60 of a second.

Gravity is an acceleration: it changes the velocity every frame, and the velocity changes the position. Each frame: $vy = vy + gravity$, then $y = y + vy$.

1 Screen Coordinates

The game window is **800 pixels wide** and **600 pixels tall**, with (0, 0) at the top-left.

a. What are the coordinates of the **center** of the window?

x: _____

y: _____

b. The player is at (200, 300) and moves **up** 50 pixels. Where is the player now?

x: _____

y: _____

2 Velocity per Frame

A ball starts at $x = 40$ and moves with $vx = 6$ pixels per frame. Show your work.

a. Where is the ball after **10 frames**?

b. How many frames does it take to reach $x = 400$?

c. If the game runs at 60 fps, how many **seconds** is that?



Game Design 3: Game Math and Collisions (page 2)

3 Delta Time

A spaceship moves at **240 pixels per second**, using `distance = speed x dt`. Complete the table.

Frame rate	dt (seconds per frame)	Pixels moved per frame	Pixels moved in 1 second
60 fps			
30 fps			

a. Another game moves its ship **4 pixels every frame** with no `dt`. At 30 fps, how many pixels per second is that? What will players notice?

4 Trace a Jump

Start: `y = 100`, `vy = -12` (negative means moving **up**), `gravity = 3`. Frame 1 is done for you.

UPDATE (EVERY FRAME)

```
1 vy = vy + gravity
2 y = y + vy
```

a. What is the player's **highest** point (smallest y)?

b. On which frame is the player back at `y = 100`?

Frame	vy after line 1	y after line 2
1	-9	91
2		
3		
4		
5		
6		
7		

Axis-Aligned Bounding Boxes (AABB)

An **axis-aligned bounding box (AABB)** is a never-rotated rectangle around an object, used to check collisions. It stores `x`, `y` (the **top-left** corner), `w` and `h`, so its right edge is at `x + w` and its bottom edge is at `y + h`. Boxes A and B **overlap** only if **all four** comparisons below are true; if even one is false, they do not collide.

THE FOUR AABB COMPARISONS

```
A.x < B.x + B.w    // A's left edge is left of B's right edge
A.x + A.w > B.x     // A's right edge is right of B's left edge
A.y < B.y + B.h     // A's top edge is above B's bottom edge
A.y + A.h > B.y     // A's bottom edge is below B's top edge
```



Game Design 3: Game Math and Collisions (page 3)

5 Check for Collisions

The player is { x: 100, y: 200, w: 32, h: 48 }. For each object, write each comparison with the numbers filled in, then **T** or **F**.

Comparison (A = player)	B = coin { x: 125, y: 240, w: 16, h: 16 }	B = spike { x: 140, y: 230, w: 20, h: 20 }
A.x < B.x + B.w		
A.x + A.w > B.x		
A.y < B.y + B.h		
A.y + A.h > B.y		
Collide? (yes / no)		

6 Debug the Collision Check

In testing, when the player walks near the bottom of the level, coins **all across the top of the screen** disappear. Remember: `&&` means AND, `||` means OR, and `&&` is worked out **before** `||` (like multiplication before addition).

```
1 function overlaps(a, b) {  
2   return a.x < b.x + b.w &&  
3     a.x + a.w > b.x &&  
4     a.y < b.y + b.h ||  
5     a.y + a.h > b.y;  
6 }
```

a. Test it with the Question 5 player and a far-away coin { x: 600, y: 50, w: 16, h: 16 }. Write T or F for lines 2 to 5, then what the function returns.

Line 2: _____

Line 3: _____

Line 4: _____

Line 5: _____

Returns: _____

b. Which line has the bug? Explain why it makes the function return the wrong answer.

c. Rewrite the buggy line correctly.