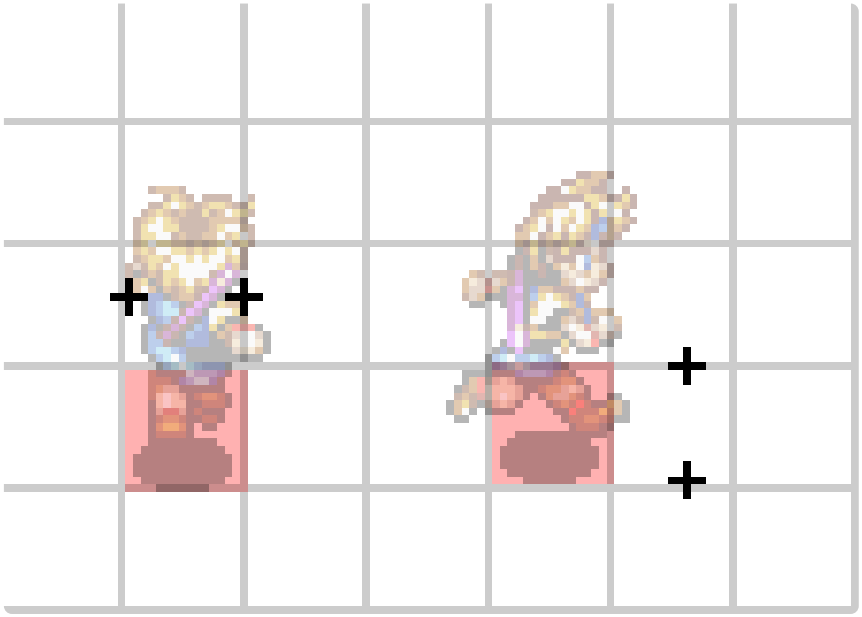
Collision

This is a tutorial that explains various simple ways to program collision in games



Separate horizontal and vertical movement.

It's often best to check for collision twice. Once for horizontal movement, and once for vertical movement.

This way, if the player is moving diagonally, and they hit a wall, they'll be able to slide along the wall instead of stopping dead in their tracks.

Plus-sign collision

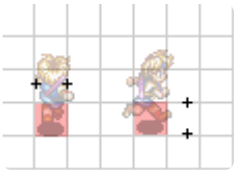
When doing collision detection, the first thing that might come to mind is to simply look at the location in front of the player, and stop if it's a wall.

The problem is that this collides like a plus sign. It's fine for tile-based movement. But walls will only block the player when they're moving toward them. So if the player walks parallel along a wall, there's nothing to stop them from halfway overlapping it.

Square collision

A better method is to collide like a square.
The key to doing this is to check in front of the CORNERS of the player's hit-box instead of checking in front of the player's center.

For example, when moving right, you would check two locations. First, you figure out where the top-right corner is, then you add the horizontal movement speed to see where that corner WILL BE on the next frame. If that location is a wall, you set the horizontal movement to Zero, preventing movement in that direction. You also check to see where the bottom-right corner will be on the next frame. Basically, if either one of those corners is about to hit a wall, you tell the player to stop moving horizontally.
If the player is moving up, you check where the top-left and top-right corners are going to be on the next frame. If either of those locations is a wall, you set their vertical movement to zero, which makes them stop before entering a wall.



Because each check involves looking at two points. I sometimes call this "2-point collision"

Use arrow keys to move the box.
Download Source

Detecting walls

One of the simplest ways to detect walls is to simply draw them in Flash and use its hitTest() function is detect them.

This is a prototype that I made for PlayShapes
Download Source

Another approach is to use a 2-dimensional array. Which is basically an array of arrays, which form a grid of variables that you can access using horizontal and vertical coordinates. This is a very fast technique, but requires the walls be tile-based and not move.

Download Source

Larger characters

Larger characters, such as Peach in the example above, may need more "corners". This is only really necessary when colliding with tiles. But it also improves accuracy when colliding with drawn walls.