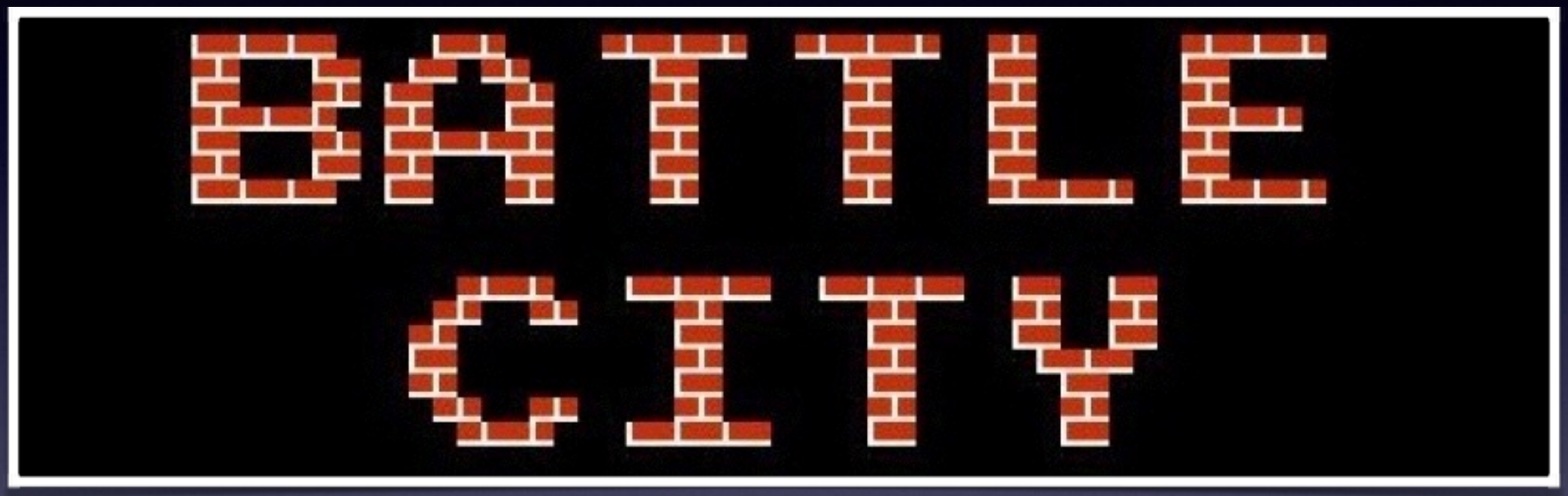
My version of

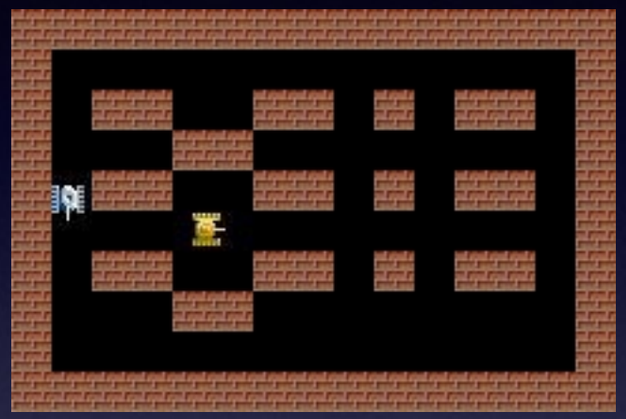


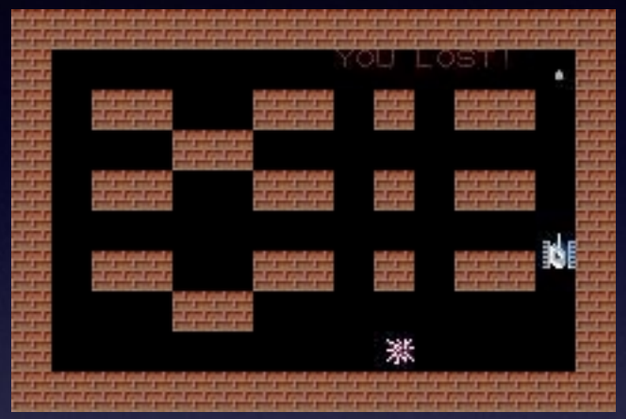
from 1985

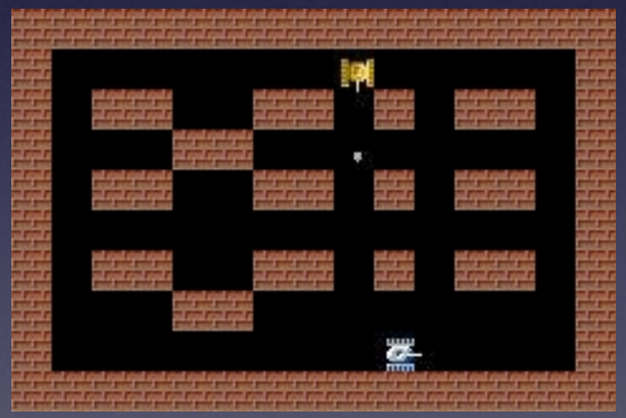
by Matthew Wallace

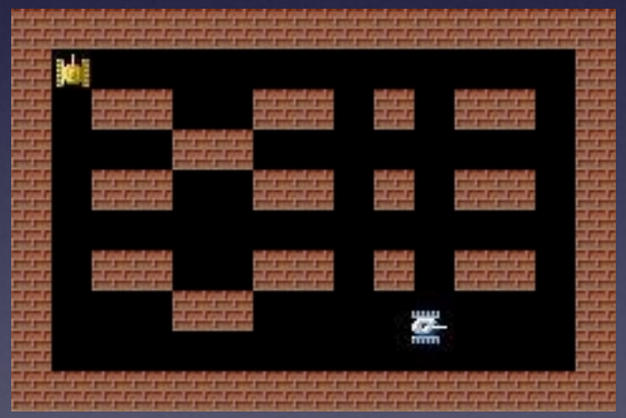
Versions Comparison

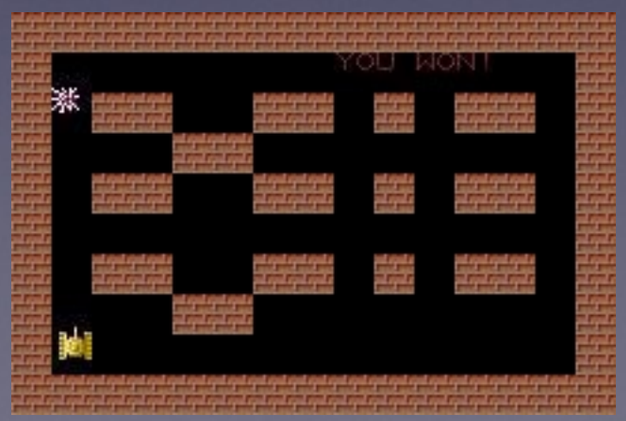
My version



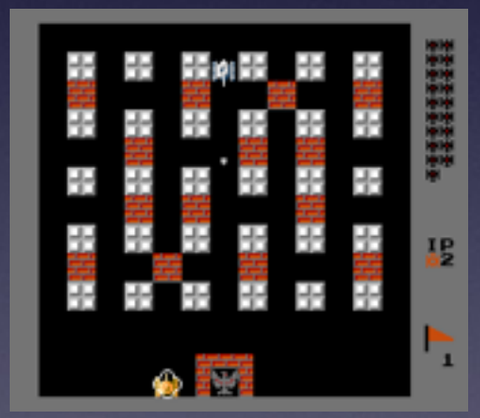








The original version



Design

In my version of *Battle City,* the player controls a tank around a map made out of brick walls and is prevented from walking through the walls. The player can press a button to shoot a bullet, which travels in a ballistic path before hitting an enemy and destroying him or hitting the wall and disappearing. The enemy is moving around the map in predefined path and shooting randomly.The game finishes when the player shoots the enemy or the enemy is shot by the player.

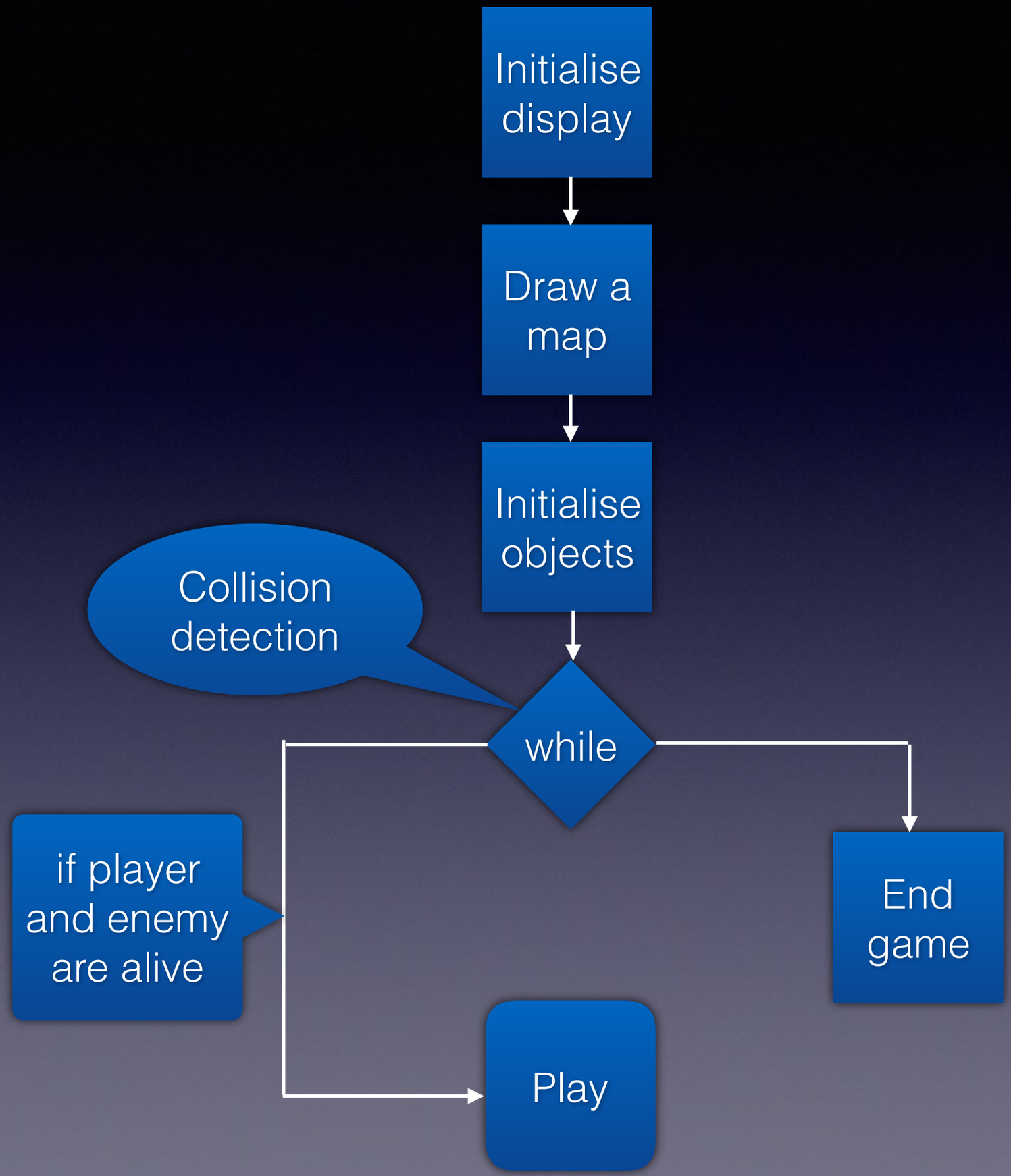
Implementation

I structured my code using:

- A *defines* file containing all my CONSTANTS
- *main* file containing a running program
- *classes* for the player, the enemy and the bullet
- *map* file containing a two-dimensional array for the map structure
- Sprite sheets for *assets* and *background* converted into files with Grid program

Main structure

- *Initialise display* function in a tiled graphics mode with tiled backgrounds and sprites
- *Draw a map* function
- Initialising objects: *player, enemy, player's bullet, enemy bullet*
- *while* game loop. Game continues while the player and the enemy are alive.
- if *enemy* is alive, he's moving and randomly shooting
- if *player* is alive it's possible to move and shoot
- Collision detection between player's bullet and the enemy
- Collision detection between enemy's bullet and the player


Initialise display
Draw a map
Initialise objects
Collision detection
while
if player and enemy are alive
Play
End game

Main features

- Collision detection
  - Player/enemy/bullet
  - Player/enemy/walls
- Movement
  - Player/enemy
  - Bullet
- Enemy random shooting

Collision detection

For the player/enemy/bullet

- Putting the player, the enemy and the bullet into a square box of their width and height.

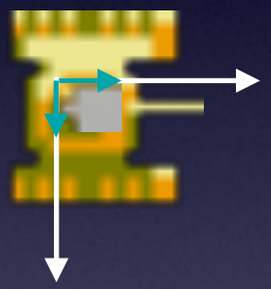
- Changing bullet’s width and height for X and Y trajectory

- Player’s and enemy’s width and height stays always the same

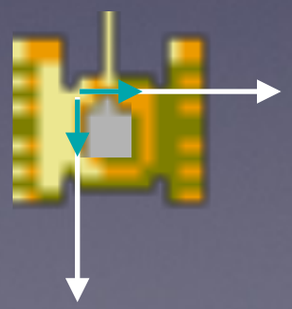
Collision detection

For the player/enemy/bullet

bullet.X < player.X + player.width AND
bullet.X + bullet.width > player.X



bullet.Y < player.Y + player.height AND
bullet.Y + bullet.height > player.Y



Collision detection

For the player and the enemy movement

Checking at which tile the player is

↓

Checking if all four tiles are EMPTY

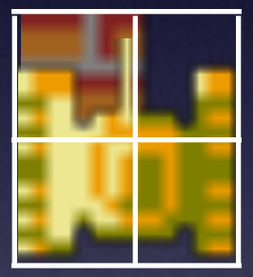
↓

if the whole tile is EMPTY moving the player

Collision detection

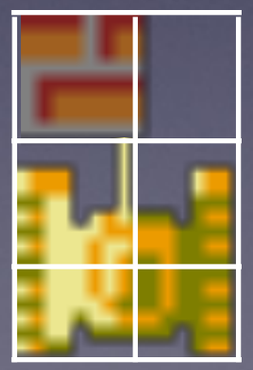
For the player and the enemy movement

- Only checking if the tile is EMPTY:



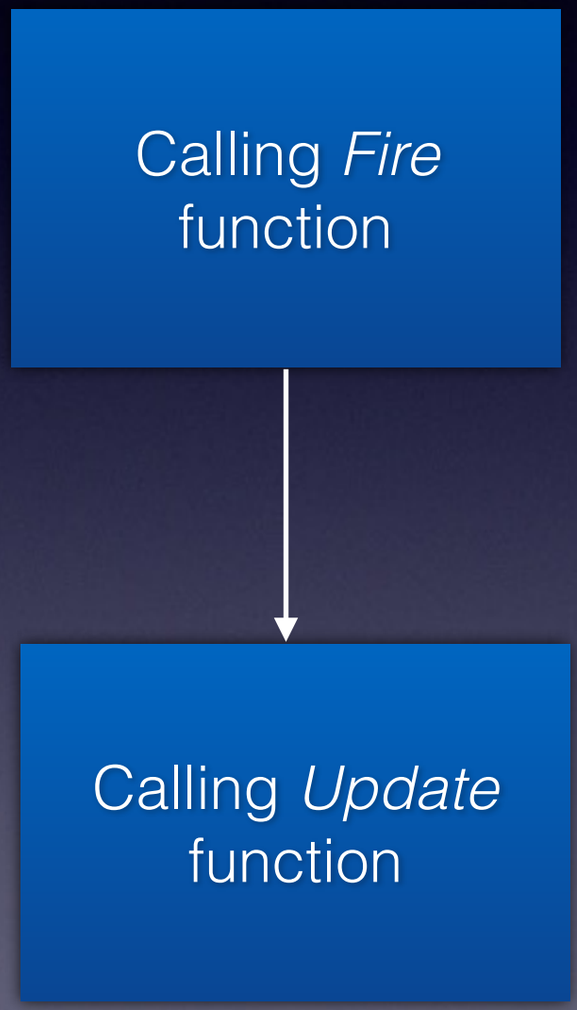
The player will move to the next tile even if all four tiles are NOT EMPTY.

- Checking if all four tiles are EMPTY:



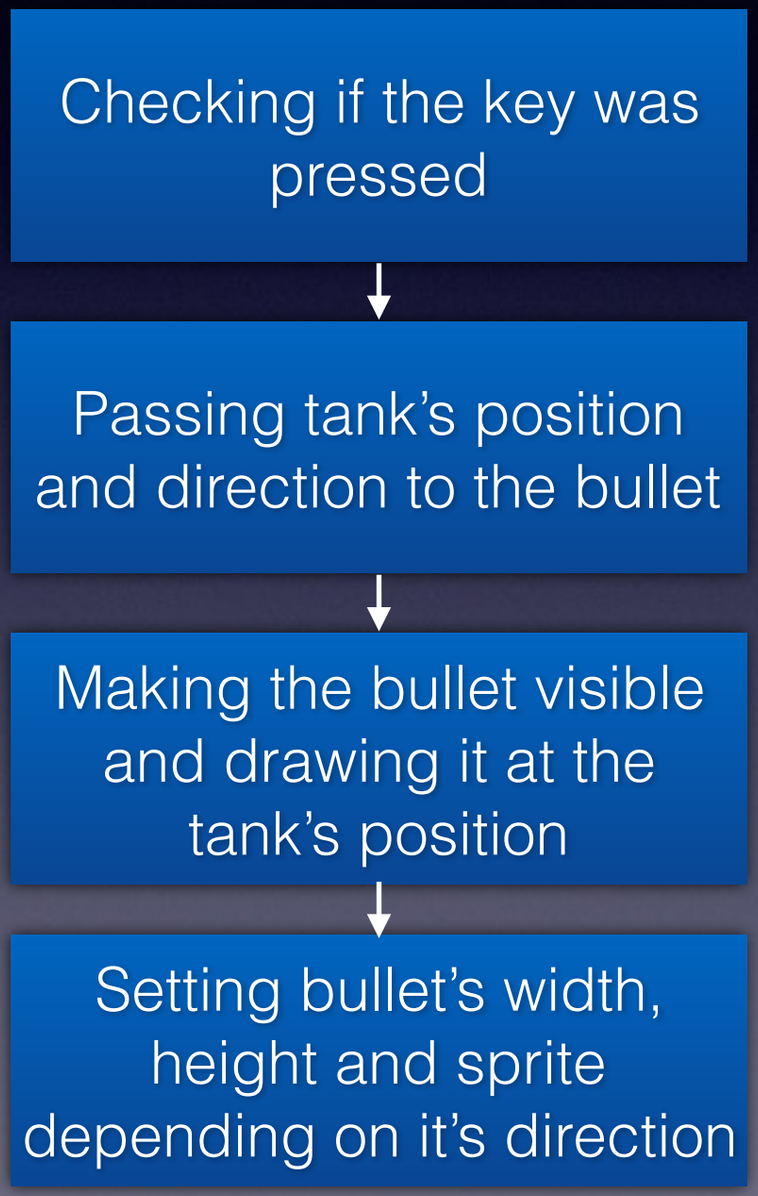
The player will move to the next tile only if all four tiles are EMPTY

Firing the bullet



Firing the bullet

Fire function



Firing the bullet

Update function

Checking bullet's position on the map

↓

Checking if the next tile is EMPTY

↓

if the tile is EMPTY moving the bullet to it

↓

or else hiding the bullet, setting it at the position (0, 0) and reseting it's coordinates to 0

Firing the bullet

The enemy shooting randomly

- Calling enemy fire function only when reminder after dividing randomly generated number by 60 equals 0

Firing the bullet

The enemy shooting randomly

if reminder after dividing randomly generated number by 60 equals 0 fire enemy's bullet

Update enemy's bullet

Changes

- Adding a spawning system for enemies
- Adding destructible walls
- Adding a scoring system