# Fort-Bishop → Game Class Layout

## 1.1 Game class methods (game.h, game.cpp)

class Game
↳void init()
↳void handleEvents()
↳void update()
↳void render()
↳void clean()

**init()** → Sets up game and game variables upon getting called. Creates **SDL_Render**, **SDL_Window**, and sets up **tilemap**, **player**, and other entities.

**handleEvents()** → Polls events (saving the event(s) to a variable accessible across multiple files), and handles most of the polled events like **SDL_Quit**.

**update()** → Updates everything by combining a bunch of smaller update functions. Every **ECS component** has an update function. Every **Entity** has an update (loops through components and calls **component→update()**). The **Manager** has an update function (loops through all entities and calls **entity→update()**). The update function handles all collisions, and other things that need to be updated that aren't ECS, for example **Game::Camera**. **Note:** this method does NOT render anything to the screen. That is done in the **Game::render()** method.

**render()** → Renders all render layers to the screen. Everything to be rendered (like entities) will be part of a render layer, for example **groupInactiveUI**. **Note:** this method does NOT update anything. That is done in the Game::update() method. It is important that the **Game::update()** method is called before the **Game::render()** method so that there is no lag.

**clean()** → When the player exits the game (with the "X" button), this method will clean up the game by destroying certain entities, getting rid of the SDL_window and SDL_render, and then calling SDL_Quit() which will quit SDL.

## 1.2 How main.cpp puts together the game

**main.cpp** can be found outside the src/ folder. The file online includes one file → #include game.h. This is the only file to have an "int main()" function. In the main function, a new Game object is created. First the init method is called. Then in a while loop the following functions are called in a specific order (refer to section 1.3). Once the mainloop has been exited

(SDL_Quit), the clean method is called. This file is quite small compared to the others, but it puts together the entire game by calling the methods in game.h & game.cpp.

### 1.3 main.cpp int main() layout

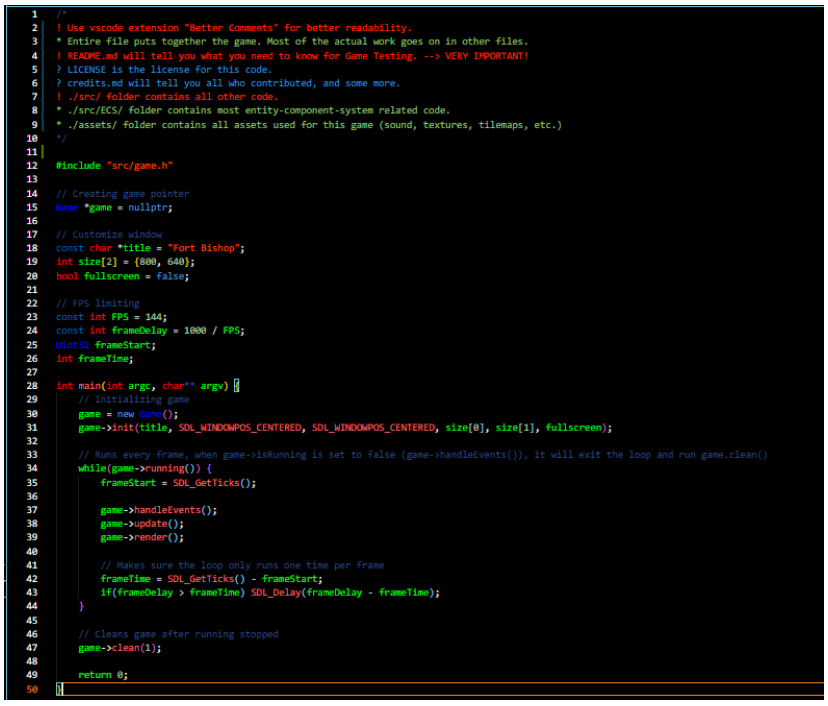
```
int main() {
        game->init(<args>);

        mainloop {
                game->handleEvents();
                game->update();
                game->render();
        }

        game->clean(<args>);
}
```

**Notes:**

- this isn't all that there is in the main function
- The **mainloop refers to the following line →** while(game->isRunning())

### 1.4 Screenshots

main.cpp



```
/*
! Use vscode extension "Better Comments" for better readability.
* Entire file puts together the game. Most of the actual work goes on in other files.
! README.md will tell you what you need to know for Game Testing. --> VERY IMPORTANT!
? LICENSE is the license for this code.
? credits.md will tell you all who contributed, and some more.
! ./src/ folder contains all other code.
* ./src/ECS/ folder contains most entity-component-system related code.
* ./assets/ folder contains all assets used for this game (sound, textures, tilemaps, etc.)
*/

#include "src/game.h"

// Creating game pointer
Game *game = nullptr;

// Customize window
const char *title = "Fort Bishop";
int size[2] = {800, 640};
bool fullscreen = false;

// FPS limiting
const int FPS = 144;
const int frameDelay = 1000 / FPS;
Uint32 frameStart;
int frameTime;

int main(int argc, char** argv) {
    // Initializing game
    game = new Game();
    game->init(title, SDL_WINDOWPOS_CENTERED, SDL_WINDOWPOS_CENTERED, size[0], size[1], fullscreen);

    // Runs every frame, when game->isRunning is set to false (game->handleEvents()), it will exit the loop and run game.clean()
    while(game->running()) {
        frameStart = SDL_GetTicks();

        game->handleEvents();
        game->update();
        game->render();

        // Makes sure the loop only runs one time per frame
        frameTime = SDL_GetTicks() - frameStart;
        if(frameDelay > frameTime) SDL_Delay(frameDelay - frameTime);
    }

    // Cleans game after running stopped
    game->clean(1);

    return 0;
}
```