Hello Jari

The program runs fine and the game looks great, good job on this. The likeness of the clone and game logic itself is but one of the requirements for the assignment, most other requirements is focusing on the construction of the program and the code itself.

And there are a lot of problems in this area unfortunately, first thing is that your project did not even compile at first because you do not have relative paths set to the include and lib folders. The program has 48 warnings most of them conversion between float and int or vice verca, but it paints a picture of you either using the wrong datatype for its purpose or simply not caring for the warnings, of which either is bad. There is not a clear separation of some of the systems from the game implementation like the requirements state, and there are no comments in any of the systems for handling different aspects of the program that I can find (rendering, audio, input and event, texture, collision, states or entities). Most of them are also hard coded with literal values so that they always have to be updated for things to work if anything new would be added, if I a new enemy was needed it would not be enough to create that specific header and cpp file, you would also have to update the Textures.cpp file with the path and name of that texture.

You do not separate declaration and definition or initialization; the header files should not have definitions and should not initialize variables this should be done in the cpp file. You tend to include more than you need in header files instead of forward declaring, you send pointers when a reference would be better. You define namespaces in one file but then disregard it by typing using of that said namespace in other files, namespaces are there for a reason.

Last thing is the clean code aspect of looking at the program from a higher level, and I am afraid that if I were to be blunt it is quite alarming because of the lack of consistency. Functions you create have far greater responsibilities than their name convey, variables are used in functions but also public and are haphazardly just changed directly or through the function based on what looks as convenience when coding. For example Block in the cpp file has a function called render that takes a SDL\_Renderer pointer, first of if you send pointers you need to check so that it is not a nullptr besides that I expect the rest to be code that renders the block. But this function does more, it loads a texture (line 151) for to me an unknown reason, it seems to perform an update on its colliders and window rect and it even adds powerups to the map.

Clearly not just rendering, half of the variables it has are public and if moved to private brings up 46 errors most of which in all honesty are used for checking values so adding a handful of get functions would solve this. But it also becomes evident that other classes are reaching into the class objects and changing internal properties. The example of this specific function and class is true for a lot of things in the project, this is just not acceptable from a clean code perspective or a game programming perspective.

Regarding the advanced points your program has no memory leaks and it has data driven entity creations and pathfinding (but it is all in a .h file and spread to be coupled with the block class). The animation is not a class and the AI does not contain different internal states or transitions between them. The program all in all needs a heavy makeover codewise for it to pass.