Index

Part One: Errors and Fixes

Description

This file contains all the bugs, errors and mistakes along with the fixes done in the Auto Battle RPG, provided by Kokku. This file is an attempt to document all the changes, why they were done and how they affect the project as a whole.

Errors

This section contains all the bugs, errors and mistakes of the code, things that stop the code from executing along with mistakes that make the game loop behave in undesirable ways.

1. **Vector: No such file or directory(in Grid.h):** simple typo, it should be vector without the capital ‘v’.
2. **Vector has not been declared AND expected ‘,’ or ‘...’ before ‘<’ token(in Character.h):** an error that occurs by not declaring ‘using namespace std’ before the class and after the libraries.
3. **‘Shared_ptr’ does not name a type(in BattleField.h):** the code didn’t include the ‘<memory>’ library.
4. **Switch quantity not an integer AND expected unqualified-id(in BattleField.cpp):** the variable choice should be an integer so the switch may work and after changing it to an integer it doesn’t need the ‘std::’ in front of the variable.

5. **No Matching function for call to ‘getline()(in BattleField.cpp)’:** this function to get the line doesn’t exist in the code and furthermore it isn’t needed, as for the ‘cin >> choice’ already does that.
6. **Conversion from ‘const char*’ to ‘int’ in a converted constant expression(in BattleField.cpp):** caused because the switch was treating the player choices (1-4) as string not as integers.
7. **Cast to pointer from integer of different size(in BatteField.cpp):** happened because classIndex, that was declared as an int was casted to an int *, so we should pass &classIndex instead.
8. **Cannot convert ‘Character*’ to ‘std::shared_ptr<Character>*’ in assignment(in BattleField.cpp):** it happens because the class Character doesn’t have the type of a shared pointer.
9. **Cannot convert ‘std::shared_ptr<Character>’ to ‘std::shared_ptr<Character>*’ in assignment(in BattleField.cpp and BattleField.h):** simply a faulty use of a pointer, only needing to remove the ‘*’ in the ‘std::shared_ptr<Character>*’.
10. **Cannot convert ‘std::shared_ptr<Character>’ to ‘Character*’ (in BattleField.cpp and Character.h):** the target from the character class needs to me a shared pointer and not a Character*, as in the BattleField.cpp code EnemyCharacter and PlayerCharacter receive a shared pointer as targets.
11. **No matching function for call to ‘std::__cxx11::list<Character>::push_back(std::shared_ptr<Character>&)’(in BattleField.cpp):** just needed to pass the memory address of the PlayerCharacter and EnemyCharacter through the function.
12. **No matching function for call to ‘Character::Character(Types::CharacterClass*&)(in BattleField.cpp):** removed the ‘*’ and the change done on item 7. of this list(the & from ‘&classIndex’) in ‘Types::CharacterClass* characterClass = (Types::CharacterClass*)&classIndex’.
13. **‘shared_ptr’ does not name a type(in CharacterClass.h):** imported the memory library.
14. **No matching function for call to ‘Types::GridBox::GridBox()’(in Character.cpp and Character.h):** had to add a ‘*’ in ‘Types::GridBox currentBox;’.
15. **Cannot convert ‘std::shared_ptr<Character>’ to ‘Character*’(in Character.cpp and Character.h):** simply change the type that Attack() receives in Character.h to a shared pointer.
16. **No declaration matches ‘void Character::Attack(Character*)’(in Character.cpp):** fixed by changing the declaration of the method Attack(Character* target) to Attack(shared_ptr<Character> target).
17. **Request for member ‘xIndex’ in ‘((Character*)this)->Character::currentBox’, which is of pointer type ‘Types::GridBox*’ (maybe you meant to use ‘->’ ?) AND request for member ‘xIndex’ in ‘((std::__shared_ptr_access<Character,**

__gnu_cxx::_S_atomic, false, false>*)(in Character.cpp): simply change the '.' to '->' in the line 'currentBox.xIndex > target->currentBox.xIndex'.

18. **Cannot convert 'Types::GridBox' to 'Types::GridBox*' in assignment(in BattleField.cpp):** added & identifier in front of '*l_front' on two lines ('PlayerCharacter->currentBox = &*l_front;' and 'EnemyCharacter->currentBox = &*l_front;').
19. **Changing various address identifiers from '.' to '->'(in Character.cpp):** due to the fact that there are a various number of address identifiers done wrong on Character.cpp, I simply put them all in this error just to simplify the documentation.
20. **No match for 'operator=' (operand types are '__gnu_cxx::__alloc_traits<std::allocator<Types::GridBox>, Types::GridBox>::value_type' {aka 'Types::GridBox'} and 'Types::GridBox*')(in Character.cpp and Grid.h):** changed the variable type in Grid.h of the grids variable from 'std::vector<Types::GridBox> grids' to std::vector<Types::GridBox*> grids.
21. **Warning: no return statement in function returning non-void(in Character.cpp):** added a temporary return statement at the CheckCloseTargets method.
22. **'printf' was not declared in this scope(in Grid.cpp):** added stdio.h so I can have access to the printf function.
23. **No matching function for call to 'Types::GridBox::GridBox()':** changed the line 'Types::GridBox* currentgrid = new Types::GridBox();' to 'Types::GridBox* currentgrid = grids[Columns * i + j];'
24. **Cannot convert 'Types::GridBox**' to 'Types::GridBox*' in initialization(in BatteField.cpp):** reverted the changes of item 18. and removed the & identifier and added the * for all the '= *l_front' lines.
25. **Request for member 'occupied' in '* l_front.__gnu_cxx::__normal_iterator<Types::GridBox**, std::vector<Types::GridBox*> >::operator->()', which is of pointer type 'Types::GridBox*' (maybe you meant to use ->' ?)(in Battlefield.cpp):** added * in front of the 'l_front->occupied = true;' lines.
26. **ISO C++ forbids comparison between pointer and integer [-fpermissive](in BattleField.c):** removed the '->Index - 1' of the line 'if(find(battlefield->grids.begin(), battlefield->grids.end(), currentBox) != battlefield->grids.end())'.
27. **'AllPlayer' was not declared in this scope; did you mean 'AllPlayers'?(in Battefield.h):** changed the AllPlayers list from <Character>* to shared_ptr<Character>.
28. **No match for 'operator=' (operand types are 'std::__cxx11::list<std::shared_ptr<Character> >' and 'std::__cxx11::list<Character>*')(in BattleField.cpp):** 'AllPlayers = new list<Character>();' isn't used on the code.

29. **Base operand of '->' has non-pointer type 'std::__cxx11::list<std::shared_ptr<Character> >'(in Battefield.cpp):** changed address identifier from '->' to '.' in multiple instances.
30. **No matching function for call to 'std::__cxx11::list<std::shared_ptr<Character> >::push_back(std::__shared_ptr_access<Character, __gnu_cxx::_S_atomic, false, false>::element_type&)'(in BatteField.cpp):** removed '*' address identifier in front PlayerCharacter and EnemyCharacter.
31. **No match for 'operator=' (operand types are 'std::__cxx11::list<Character>::iterator' and 'std::__cxx11::list<std::shared_ptr<Character> >::iterator')(in BattleField.cpp):** the list iterator must be from shared_ptr<Character> and not from only Character.
32. **'class std::shared_ptr<Character>' has no member named 'StartTurn':** placed '**' in front the 'it->StartTurn(grid);' line.
33. **base operand of '->' has non-pointer type 'std::__shared_ptr_access<Character, __gnu_cxx::_S_atomic, false, false>::element_type' {aka 'Character'}:** changed the '->' to '.' on the 'it->StartTurn(grid);' line.

Adjustments

This section contains all the things that I adjusted, viewed that I thought needed adjustments simply for good practices or for optimizing the code.

1. **Non-existence of a makefile:** the code doesn't have a makefile to execute all the '.c' and '.h' files, with the existence of the makefile it makes easier for the user and the developer to execute everything without needing to know the code (which part should execute to get the full code(encapsulation)), and they don't need to type all the bash code every time.
2. **Order of the includes:** there isn't a fixed "good way" to include all the libraries of C++, but I prefer using all the system ones at the top followed by the personal '.h'.
3. **Ident inside the classes:** inside some classes the indent isn't done properly, so I had to fix it following good practices.
4. **Useless spaces and spacing:** through the code are some spacing and line breaks that fill the code with useless spaces and spacing, I removed them to make the code more easy to read.
5. **Missing spacing of problems with line breaks:** had to include some spacing and line breaks where needed to make the code easier to read.
6. **Inconsistency of the opening and closing of the brackets:** the brackets' various instances are opened in different ways(along the method or under it), so I put them under the method to keep consistency.

7. **Removing unnecessary parentheses and other punctuation:** in some lines of code there are unnecessary parentheses and other punctuations that are not needed.
8. **Fixing typos:** fixing typos on the code, such as ‘charcater class.
9. **Added some line breaks on the code:** some prints don't have ‘\n’, so the text remains clumped on the terminal.

Fixes

This section contains all the corrections done in the code, bugs and mistakes to optimizations done on it.

1. Fixed a type done to the vector library.
2. Added ‘using namespace std’ to fix the vector error.
3. Added the memory library, so the code could recognize the shared_ptr.
4. Changed the variable type of choice to an integer.
5. Removed the ‘std::’ that was in front of the choice variable.
6. Removed the ‘std::getline(std::cin, choice);’
7. Fixed the switch by removing the quotes from the numbers, so it used integers and not string.
8. Added the address of the variable classIndex instead of only the variable.
9. Changed the declaration of the EnemyCharacter of ‘new Character(enemyClass)’ to ‘std::make_shared<Character>(enemyClass)’, to create a shared pointer.
10. Removing the ‘*’ in ‘shared_ptr<Character>* EnemyCharacter;’ .
11. Changing the target variable type from Character* to shared_ptr<Character>.’
12. Changed the AllPlayers->push_back to AllPlayers.push_back.
13. Added the memory address of the PlayerCharacter and EnemyCharacter through the push_back function into the AllPlayers list.
14. Removed the change done on item 8. and all the ‘*’ from ‘Types::CharacterClass* characterClass = (Types::CharacterClass*) &classIndex’
15. Added the memory library in CharacterClass.h.
16. Added ‘*’ to the line ‘Types::GridBox currentBox;’.
17. Change the target type from Character* to shared_ptr.
18. Fixed the declaration of the Attack() method.
19. Changed the ‘.’ to ‘->’ in the line ‘currentBox.xIndex > target->currentBox.xIndex’.
20. Added & identifier before the *l_front.
21. Added a bunch of ‘->’ identifiers in change of the ‘.’ identifiers.
22. Added a temporary return statement to make the code work.
23. Added the stdio.h library.
24. Reverted the item 18 and removed &, adding * instead.
25. Added * in front of the ‘l_front->occupied = true;’ lines.

26. Removed '-> Index - 1' of the 'if(find(battlefield->grids.begin(), battlefield->grids.end(), currentBox) != battlefield->grids.end())' line.
27. Fixed the GetRandomInt(int min, int max) function.
28. Changed the switch statement into a if-else statement on GetPlayerChoice().
29. Changed the type of AllPlayers list to shared_ptr<Character>.
30. Removed the AllPlayers = new list<Character>();
31. Changed the address identifier from '->' to '.'
32. Removed the '*' identifier in front of PlayerCharacter and EnemyCharacter.
33. Changed the iterator type from Character to shared_ptr<Character>.
34. In the 'it->StartTurn(grid);, changed the '->' to '.' and added '**' in front of it.
35. Fixed StartTurn() method.
36. Added winning and losing messages for the HandleTurn() method.
37. Added a way to wait for the user to give an input to pass the turn.
38. Modified the StartTurn() method to use a getchar() to not show the user the second turn along with the first.
39. Created the Character::Attack() method.
40. Removed the Character::Die() method.
41. Altered the Character::TakeDamage() method, so it does indeed reduce the health and sets health = 0 so the handle turn method checks if the player died.
42. Removed the argument target from Attack(), since it already knows its target.
43. Implemented the Character::CheckCloseTargets(Grid* Battlefield).
44. Removed the canwalk argument from WalkTo(), since we can check through code if the position is empty.
45. Passed the Grid* battlefield and the int Index as an argument to WalkTo(), since the StartTurn() method has a lot of lines of code that can be swapped by creating a method that calls them separately.
46. Changed the StartTurn() so it can be easier to read.
47. Implemented WalkTo() method.
48. Added an way to tell how many characters there are in the board from both teams.

Part Two: Functionalities and Project Description

Project Description

This project is an Auto Battler Rpg, where:

1. Firstly you choose a class for two playable characters, the class can be a Paladin, Warrior, Cleric or an archer.
2. After that the classes for your enemies are chosen, and then the players and enemies are positioned on the board.
3. The game shows you the board and executes all the actions from all the characters that exist in the board, where the game will tell if they attacked or moved, the health that the attacked character has left and where the character moved.

4. Then the user will press any key to continue the game and it continues until the player team or the enemy team dies, giving a winning message if the player survived or a losing one if not.
5. The game can be played multiple times!

Functionalities

This section contains all the functionalities implemented on the project.

Battlefield.cpp and Battlefield.h:

1. GetRandomInt(int min, int max) in BattleField.cpp: generates a random integer number, based on the range passed as arguments.
2. GetPlayerChoice() in BattleField.cpp: gets user input of a class that they want to choose, showing the choices on the screen.
3. StartTurn() in BatteField.cpp: shuffles the list of players and goes through all of them to make sure they made their actions.
4. HandleTurn() in BattleField.cpp: checks if the player or the enemy died, if the player died he lost, if the enemy died the player won and if the two conditions didn't happen then it waits for an input from the user to pass the turn.
5. Battlefield::Battlefield() in Battlefield.cpp: constructor of the class Battlefield.
6. Setup() in Battlefield.cpp: let players choose their character, create enemies and start the game.
7. CreatePlayerCharacter() AND CreateEnemyCharacter() in Battlefield.cpp: creates characters for the player and enemy, doing the initial configurations.
8. StartGame() in BattleField.cpp: alocate the characters on the battlefield and calls the first turn.
9. AllocatePlayers() AND AllocatePlayerCharacter(int characterIndex) AND AllocateEnemyCharacter(int characterIndex) in Battlefield.cpp: alocate the characters in the battlefield.
10. Battlefield.h: contains the information about the battlefield.

Character.cpp and Character.h:

11. Character::TakeDamage(float amount) in Character.cpp: the function handles the damage that the character takes, along with setting his health to zero if he died.
12. Character::Attack() in Character.cpp: handles the attack that the character does, along with removing the damage amount from the target.
13. Character::CheckCloseTargets(Grid* Battlefield) in Character.cpp: checks if the enemy of the character is in close proximity to the character.
14. Character::WalkTo() in Character.cpp: handles the movement of the character through the battlefield.
15. Character::StartTurn() in Character.cpp: checks if the character will attack or walk on this turn.

16. Character::Die() in Character.cpp: sets the health of a character to 0 and turns the position in the grid to false.
17. Character.h: has the informations of the character.

Grid.cpp and Grid.h:

18. Grid::Grid(int lines, int columns) in Grid.cpp: constructor of the class Grid, contains the informations to build the grid.
19. Grid::drawBattlefield(int Lines, int Columns) in Grid.cpp: draws the Grid based on the informations contained in the constructor.
20. Grid.h: contains all the information about the grid that forms the battlefield.

Types.cpp and Types.h:

21. Types.cpp and Types.h: contains the information about the classes of the characters and the information contained in the cells of the battlefield.

Autobattle.cpp:

22. Autobattle.cpp: executes the game.