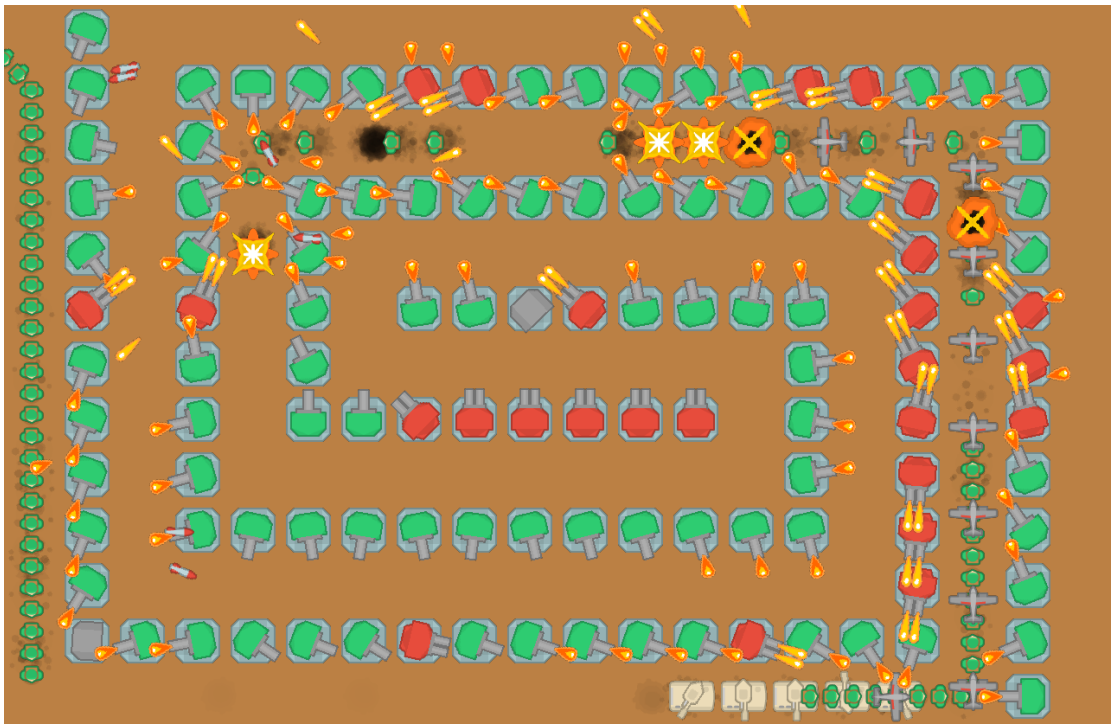
Mini Project #2

Tower Defense Game

Due: 2020/06/07 (Sun.) 23:59

Problem Description

Read the source code and finish the implementation of the Tower Defense game. Think on the list of questions below and answer them in TA's demo time.



In the playing scene, press key 0-9 to change the speed multiplier; press Q, W, E to perform quick select on different turrets; click on the empty spaces in the map to place the selected turret; press TAB to swap to debug mode. There is a hidden cheat sequence that can increase your money and nuke the entire map.

Coding Requirements (Finish the Game)

1. Add the Starting scene and start button. (0.5%)
    a. The button's image should change on mouse enter/leave.
    b. Switch to the stage selection scene when the button is clicked.
2. Add 1 new turret, 1 new enemy. (1%)
    a. Add a new turret that can be placed and will automatically attack enemies.
    b. Add a new enemy that can follow the path and die.
    c. The new turret and enemy cannot be the same as the ones in the template.

You must use different images and implement different behaviors. (e.g. turrets that can freeze or slow down enemies, enemies that will speed up or turn invisible under certain conditions)

3. Enemy pathfinding. (BFS) (1%)
    a. The enemies require pathfinding function to move toward our base.
    b. Try to implement a simple BFS to calculate the distances between any block and our base. The pathfinding should work after BFS is correctly implemented.
4. Add volume control slider. (1%)
    a. Implement a slider in the stage selection scene to support easy adjustment of background music and sound effects.
    b. The volume control of BGM and SFX should be independent.
5. Fix the bug in the Win scene. (0.5%)
    a. The game may crash when the player wins.
    b. Try to employ what you learned to find out why the game might crash. If the game does not crash during executing does not mean the code is bug-free. Visual Studio's debugger is especially useful for detecting such bugs.
    c. Make use of the tools in your IDE such as Stack Trace, Log, Watch variable, Breakpoint (step in / step out), etc.

Questions

If your answer is ambiguous during the demo, TA will ask more details regarding that question to validate your understanding. If the TA cannot understand what you wish to convey, the scores are graded subjectively by the TA in terms of your performance.

A. **Easy Questions** (2%, TA will randomly pick 2 questions below)
    1. Explain the concept of "Encapsulation". Show a short code snippet to illustrate why "Encapsulation" is useful and what might happen if we do not use it.
    2. Explain the concept of "Inheritance". Show a short code snippet to illustrate why "Inheritance" is useful and what might happen if we do not use it.
    3. Explain the concept of "Polymorphism". Show a short code snippet to illustrate why "Polymorphism" is useful and what might happen if we do not use it.
    4. What is the difference between Procedural Programming and Object-Oriented Programming? Name at least three advantages of OOP.
    5. In what condition should we call the base class function from a derived class function? In what condition can we omit the call? Elaborate them on the constructor, deconstructor, and other general overridden virtual functions.
    6. How does it feel to add a new object (e.g. Scene, Button, Turret, Enemy, …) in

OOP style instead of Procedural Programming style? In your opinion, which coding style is better?

7. What are the advantages of performing a reverse BFS, starting from our base instead of starting from each enemy? What will happen if we replace BFS with DFS? Can pathfinding be achieved through multiple forward BFSs? (Multiple BFSs starting from each enemy)
8. The header of the slider declares three Engine::Image objects. How do you utilize them to implement the slider? If you do not use them to implement the slider, how can you make use of them by reusing the methods in the Engine::Image class?
9. The game crashes when the player wins. What tools and techniques did you try to find the bug? Why does the bug crash the game? What have you learned through this debugging experience?
10. What is the hidden cheat code sequence in the game that can increase your money and nuke the entire map? How do you find it? How does the template code detect the entered cheat code sequence?
11. When there are a certain number of enemies in the "danger zone", the speed multiplier is forced to be lower than one and an intense BGM starts playing. What is the filename of the BGM? What will happen to the intense BGM if the player gets through the current crisis but immediately encountered another crisis?
12. When will the function main return the exit value zero?

B. Medium Questions (2%, TA will randomly pick 2 questions below)

1. When the value of the Slider changes, it fires a callback. Which code file contains the code of the callback function? How does that function change the BGM / SFX audio volumes?
2. In stage 2, we can build a maze through turrets, forcing enemies to take more time before reaching our base. However, there are some limits on where the player can build the turret. What are the limits and how are the limits implemented?
3. How does the game parse the file containing the map and the file containing the enemy sequence? How are the enemies generated according to the timer ticks?
4. What did the template do to achieve different speed multipliers? When the speed multiplier is zero, can the player build new turrets? Why or why not?
5. How do the homing missiles of the Missile Turret follow the enemy? Explain the math behind the homing missiles.

6. The homing missiles of the Missile Turret automatically select a target enemy to follow. What happens when the target enemy dies or the missile explodes? Does the template perform extra bookkeeping code to deal with these situations? Explain why iterators are used in such a way. If we reuse the enemy instance without deleting it upon dying, can we remove the bookkeeping code? Note that we can reuse the instance by simply adding a flag, indicating whether the instance is currently active. (kind of like the player's bullet in last semester's final project hackathon)
7. In the Play scene, multiple Engine::Groups are used for storing different objects. What will happen if we remove the groups and add all new objects directly into the scene?
8. The public change scene function of Engine::GameEngine does not change the scene immediately but waits until the next update. What might happen if the scene is changed immediately when the public function is called?
9. There is a deltaTime parameter in the update function, what are the benefits of using this parameter? What might happen if we remove the threshold of this parameter in Engine::GameEngine?
10. In Engine::Group, there are two linked-lists, storing objects and controls respectively. Can we simplify this into a single linked-list? What are the advantages and disadvantages of storing them in different linked-lists?
11. Engine::Group is a class derived from Engine::IObject, while storing a linked-list of pointers to many Engine::IObjects. Why do we store the pointer to Engine::IObjects instead of directly storing the value of Engine::IObjects.
12. Can we put all members in the Engine::Sprite class into Engine::Image class to combine them into a single class? What are the pros and cons of separating them?
13. In Win scene, the code allocates space by the new statement but does not delete them in the same file. Is this a memory leak issue? Why or why not? If the memory leakage issue exists, how can we fix it?

C. Hard Questions (1%, TA will randomly pick 1 question below)

1. What is Memory Leak? Can the invention of Smart Pointers reduce this problem? How are Smart Pointers implemented and what are the benefits of using them? Elaborate on the difference between std::unique_ptr, std::shared_ptr, and std::weak_ptr.
2. Many high-level languages forbid the usage of raw pointers. Does it mean that raw pointers are deprecated in this day and age? Come up with three situations when raw pointers are necessary.

3. Garbage Collection is a clever concept introduced in Lisp and become widely popular in higher-level programming languages such as Java, C#, Python. What does the code in Engine::Resources do? What are the benefits and drawbacks of automatically managing the resources?
4. Crash Reports are designed to send logs with detailed information to the developer, allowing them to analyze and prevent the program from crashing again. If you want to implement such a system, how can Engine::LOG help?
5. Try-Catch statements can intercept the error thrown by the inner code. Why are Try-Catches preferred instead of error codes in modern programming? List out five advantages of the Try-Catch statement.
6. There are multiple std::binds and std::functions in the template code. What are them for? How do they work? Also, elaborate on the difference between a normal function pointer and a member function pointer.
7. Factory Method can be seen as a virtual constructor while Template is a feature for classes to deal with generic types. Which concept suits better to simplifying the instantiation of different turrets? Furthermore, which of them can achieve constructing and deconstructing each scene on scene change? (In the current template we keep a single instance of each scene for simplicity)

D. Bonus Questions (At most +2%)

1. There are some parts in the code that can be optimized, be more structurally designed, or even have some minor bugs. Point out the issues and provide a detailed solution in the bonus report.
2. Your thoughts will be passed on to the TA who wrote this template, and the bonus score will be graded subjectively.
3. If you have no detailed thoughts on the template, do not hand in the bonus report. No bonus points will be given if we cannot understand your solution.

Program Submission

1. Please use Allegro & C++ to finish the game.
2. Your source file must be named as **"**`<Student_ID>_project2.zip`**"** and please make sure that all characters of the filename are in lower case. For example, if your student id is `108062000`, the name of your program file should be `108062000_project2.zip`.
3. Please only include the **"**`.cpp`**"** files, **"**`.hpp`**"** files and new resources in the zip file. The resources files of the template are too large for iLMS so do not zip the template resources files along with your code.
4. 0 points will be given to Plagiarism . NEVER SHOW YOUR CODE to others and

you must write your code by yourself. If the codes are similar to other people and you can't explain your code properly, you will be identified as plagiarism.

Report

1. Elaborate on the answer in the question list. The report is not graded directly, but indirectly through TA demo. So, make sure you have enough figures to help you explain your method to TA.
2. You can only refer to your report during the demo. Referring to your code is not allowed, so make sure you have taken a screenshot of the important code snippets.
3. The report filename must be "`<Student_ID>_project2.pdf`" and please make sure that all characters of the filename are in lower case.

Bonus Report

1. The bonus report filename must be "`<Student_ID>_project2_bonus.pdf`" and please make sure that all characters of the filename are in lower case.

Grading Policy

1. The project accounts for 9 points of your total grade.
2. You must submit both your source code and report. Remember the submission rules mentioned above, or you will be punished on your grade. No late submission is allowed.
3. The bonus score for the bonus questions can exceed the total 9 points.

- Finish the Game +4 points
- Easy Questions +2 points
- Medium Questions +2 points
- Hard Questions +1 point
- Bonus Questions +2 points