Miniclip - Bejeweled Test

By Loupas Thanos

Introduction

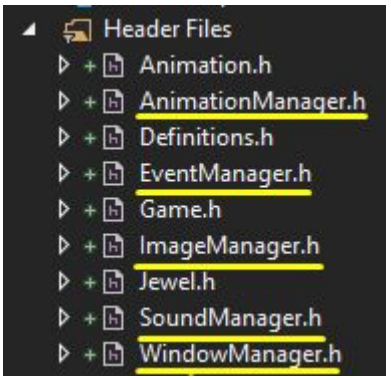
This is a complete guide for the C++ ,SDL project the way of thinking and the Implementation of it. The gameplay is very simple. The user can try and swap the jewels either by drag and drop or by clicking one and then the other. The aim is to have 3 sequential (or more) jewels. When a sequence is destroyed , new jewels are generated randomly. When the timer ends GameOver is displayed and after a few seconds a new jewel set appears. The general traverse logic in my Jewel Array follows the Cartesian coordinate system. This means that in JewelArray[x][y] , x is the position in the line and y the number of the line (in a normal array we do the opposite). This happens for array to screen coordinate alignment. During matrix init , most times we spawn 2 same color neighbors in order to create more possible sequences. No different levels of difficulty were used , but every time game ends a new map is created with the same logic.

Apart from this guide there is comment section for every function in the header files. The executable folder is MiniclipTestLoupas\x64\Release\MiniclipTestLoupas.exe . Please , do not judge harshly the lack of organisation in subfolders hierarchy (which does not exist) in my scripts images and sounds. In Unity projects and when i used IntelliJ rider or Idea there was always perfect organizing. Now with VS I had trouble make the .exe script run in the Release folder when I created subfolders and organized them.

Main

This application consists of some managers which are the “Game engine” of our application. This means that we use these managers in order to handle different operations of our game. We need these scripts in order to create and provide a basic API for the use of SDL functions and organise them.

These are :



Animation Manager

Provides the class structure and functions for creating new animations and performing them. We use the Animation class defined in another header which provides a class with screen position and functions for moving from one point to another. In the manager we perform the movement or destruction animations. I use a Queue to handle the animations , in order to get advantage of the First In First Out attribute .

Sound Manager

Provides the class structure and functions for creating new sounds and performing them . We have 2 types of sounds , music and sfx. Music is the background music and sfx is the sound effects during the destruction of the sequences.

Image Manager

This is responsible for loading game images for our objects. I load and store once every texture in a map which consists of an Enum (BoxImages) which is the type of our image (background , black , pink), and the actual SDL texture. This way we can apply the textures in our game logic part without to make the application memory heavy at runtime.

Window Manager

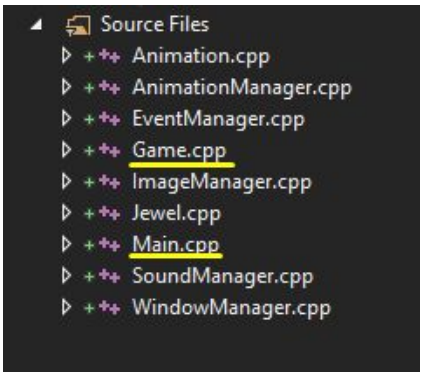
This manager is responsible for scene rendering. It initializes the screen window , holds the array of our game objects , and during the update in our main game loop renders our objects.

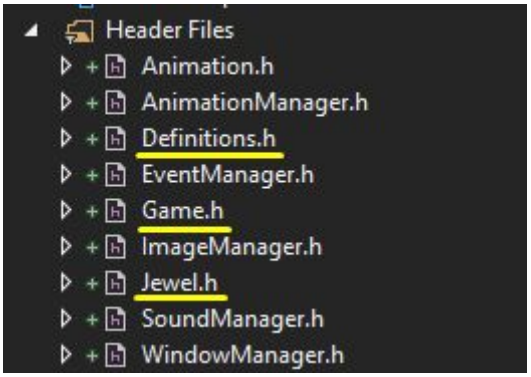
Event Manager

The event manager is the main game logic script. It is responsible for every event and action happening in our game. This means that for every jewel switch we calculate if it possible to happen , delete sequential jewels , create new ones and spawn them on the right positions and apply these changes both on the array and the animation section.

Apart from these headers , there are a few **additional ones**.

These are:





Definitions.h

It is a small header file which consists of some global variables such as Window Size and Array Size. These need to be used in multiple header files and cpps.

Game.h

This file is the one responsible for initialising, coordinating and using the managers mentioned in the section above. The Game.cpp has the Main **Game Loop** where events , updates and rendering are called.

Jewel.h

The Jewel Class is our class for handling the box game object. It consists of the array positions , the screen positions , the attribute canDrag for event handling purposes , texture of our Jewel and functions for re-calculating positions.

Timer.h

The timer class is a timer , initialised with SDL_ttf which is the library for displaying text in SDL. It provides a basic timer which lasts 30 seconds , and when it reaches 0 our game is on Game Over state. Then the timer stops rendering and the user can't click any jewel

Main.cpp

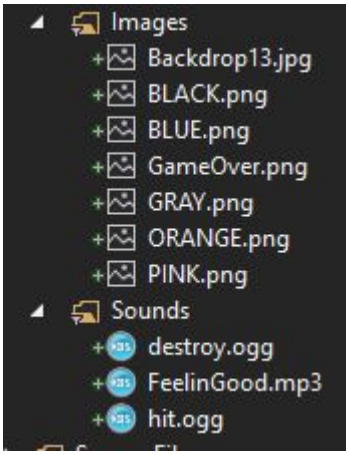
```
int main(int argc, char ** argv)
{
    Game* game = new Game();
    game->initManagers();
    game->initGame();
    game->startGame();

    return 0;
}
```

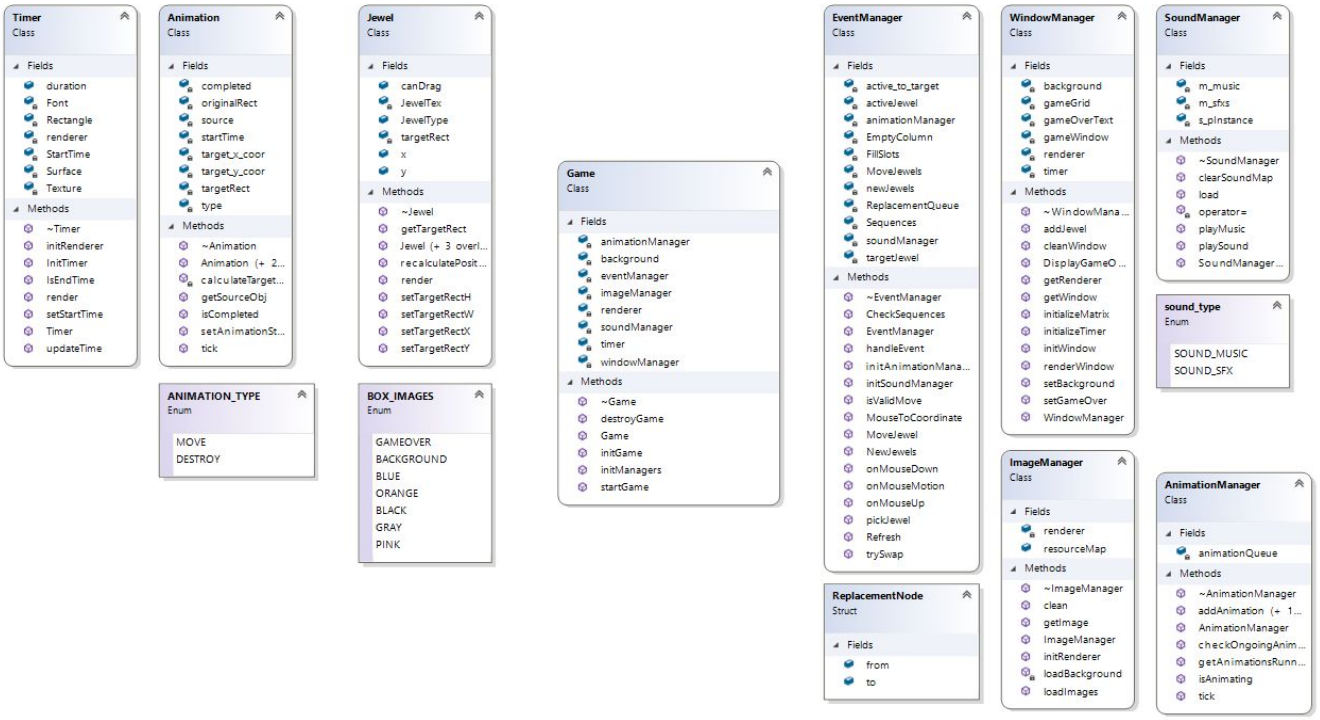
Simple and clean , creates a new game instance , initializes the managers , initializes the game and starts the game.

Additional folders :

These are my resource folders where images and sounds are stored.



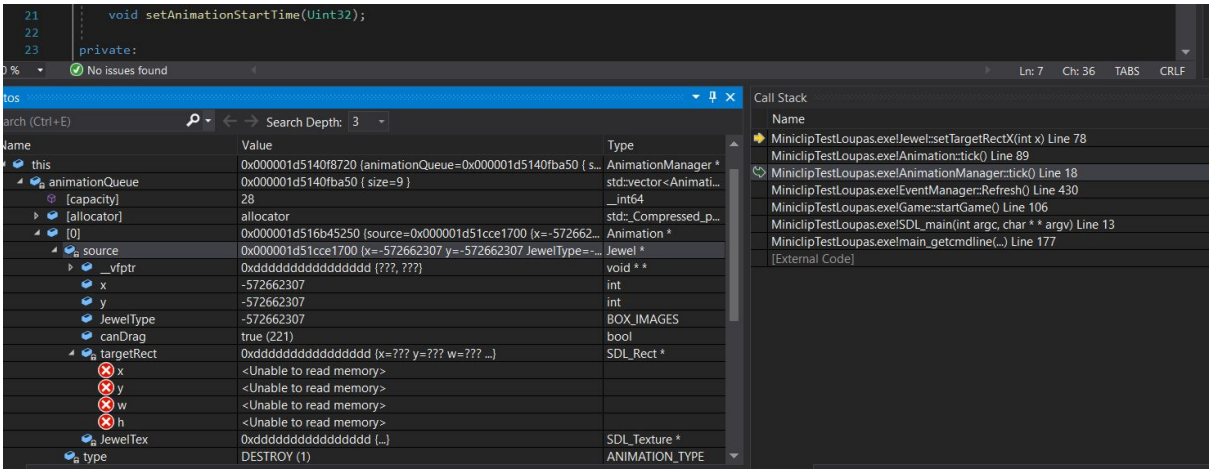
Class Diagram



Possible Bug (fixed , you can avoid this section)

There is a bug which i located and i fixed but I am writing this down just in case. Probably it won't but if it does run the application again. It was happening random on runtime and not always. Sometimes I played for 1 minute the .exe without any problem. You can notice that on my video. It appeared to be on the Animations , regarding only the Destroy animations. It caused access violation and crashed the game. The animation manager consists of animations. The animation class , apart from other attributes, consists of a Jewel* Source object. which is the object animating. Although the animation of destroy appears to happen normal on the game , there were some garbage in my animation vector.

Solution : An if check in the refresh function , regarding animation queue. It was a function scheduling issue , due to asynchronous cpp calls.



References

This is the list of sites which proved very helpful for the development .

-Lazy Foo productions
-GameDev.net
-StackExchange
-kenney.nl (loyalty free sounds sfx)
-PurplePlanet (loyalty free background music)