**University of Ontario Institute of Technology**

**INFR 2820U: Algorithms & Data Structures**

**GDW Report**

**Randy Almeida – 100551116**

**Raymond Dee – 100578667**

**Kenneth Ho – 100583602**

**Simon Pichl – 100579423**

**Cameron van Velzen - 100591663**

**Required:**

* Classes (Encapsulation)
* Polymorphism
* Inheritance
* Sorting, Searching or Additional Data Structure
* Exception Handling

**Implementation:**

**Classes:**

We use an abundance of classes, and below are the best examples of these.

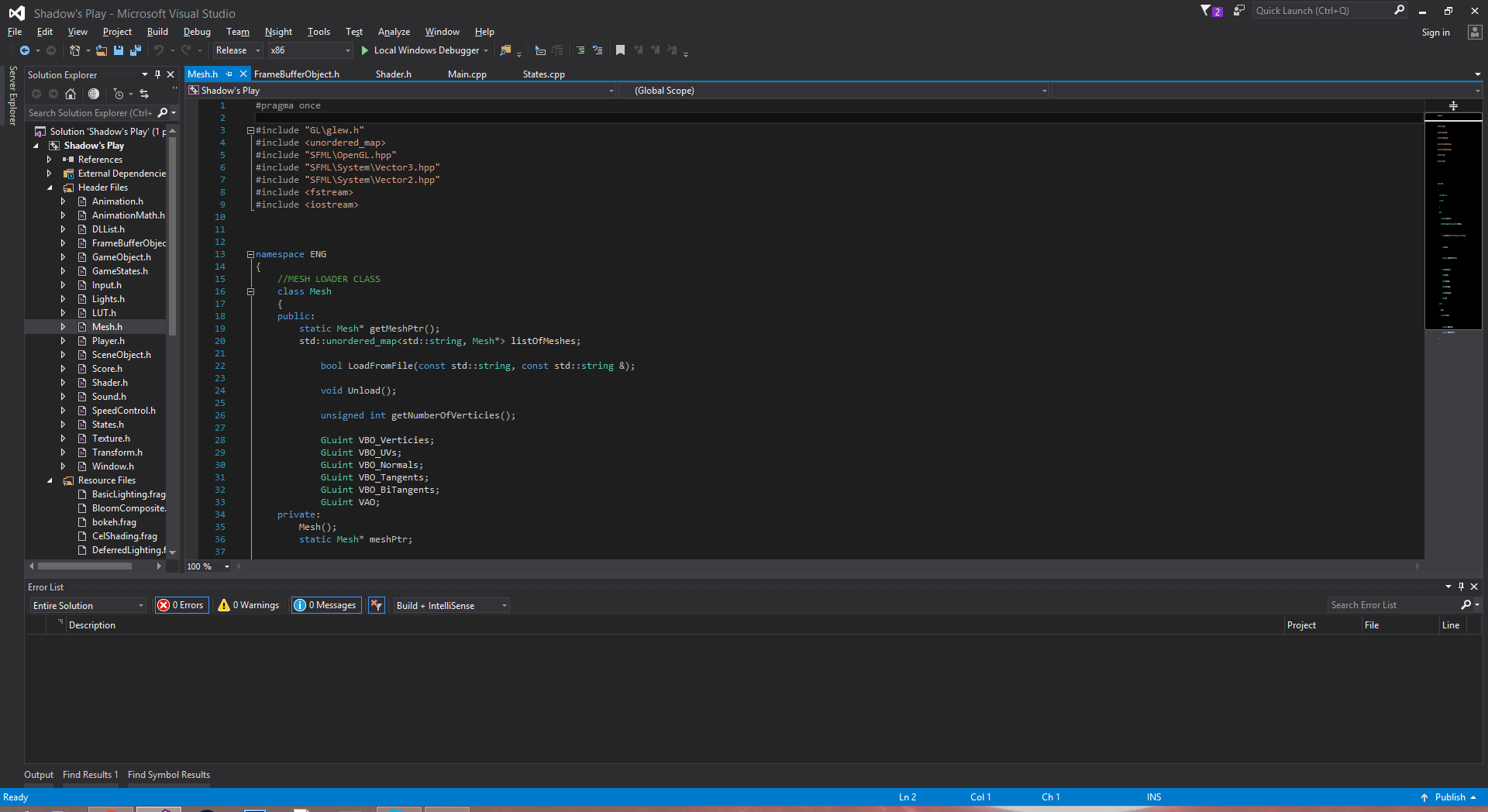
****

Figure 1: Mesh Class in the header file Mesh.h

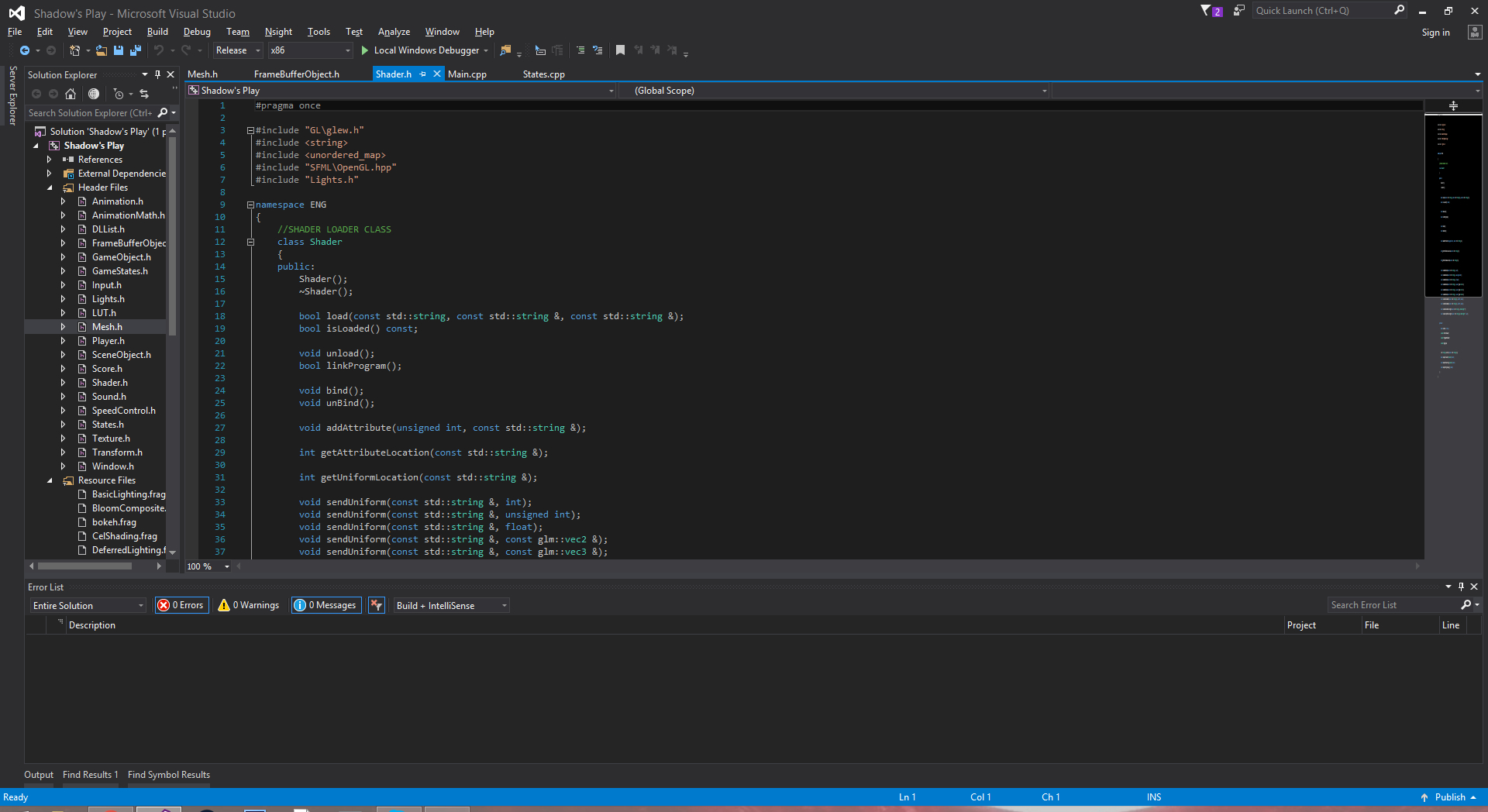
****

Figure 2: Shader Class in the header file Shader.h

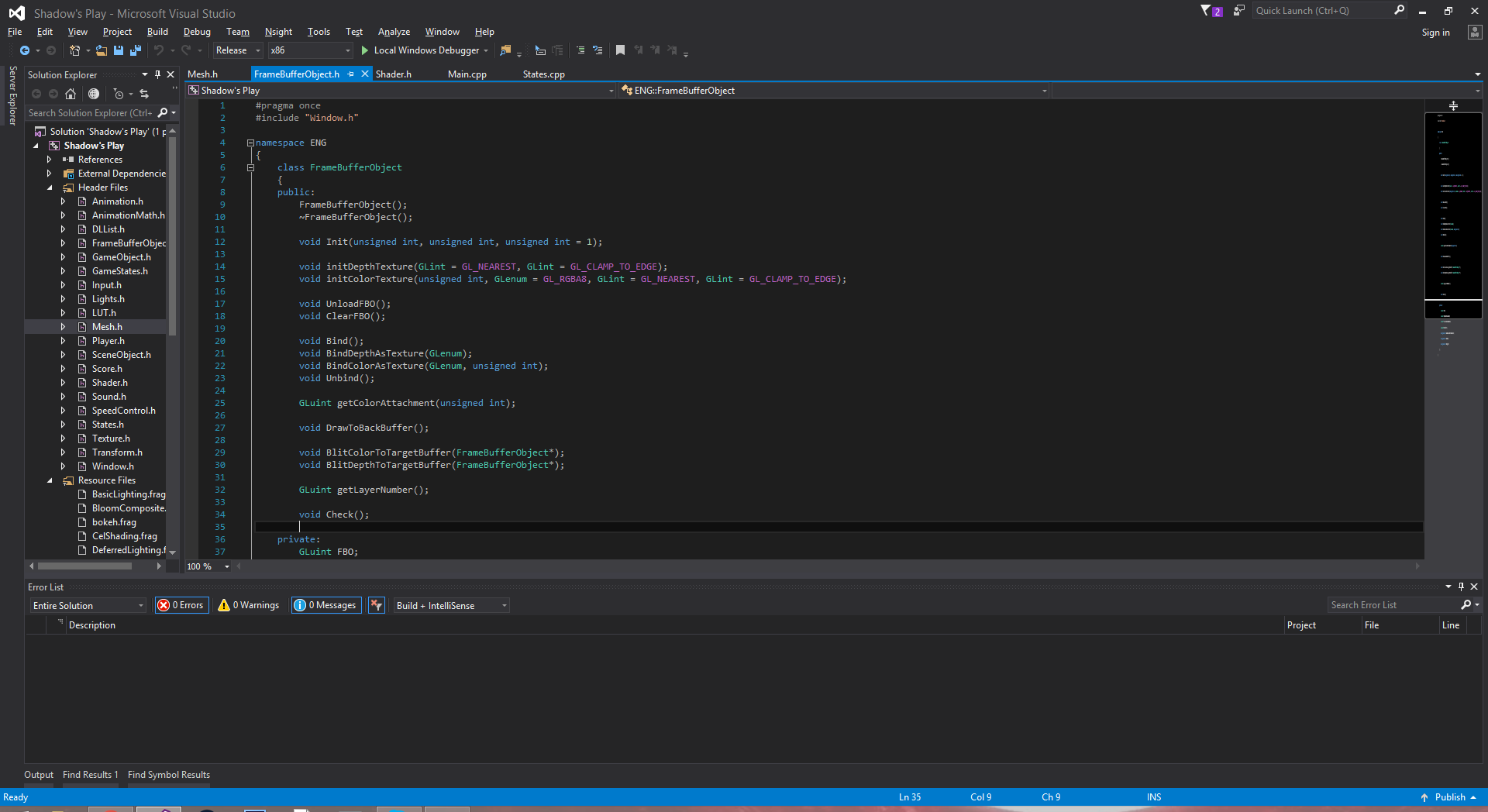
****

Figure 3: Framebuffer Object class in the header file FrambufferObject.h

**Polymorphism:**

Below are code snippets of our Game State Manager. It uses a polymorphic structure to ensure switching between game states is very fast. We create the polymorphic states and manager in Main.cpp.

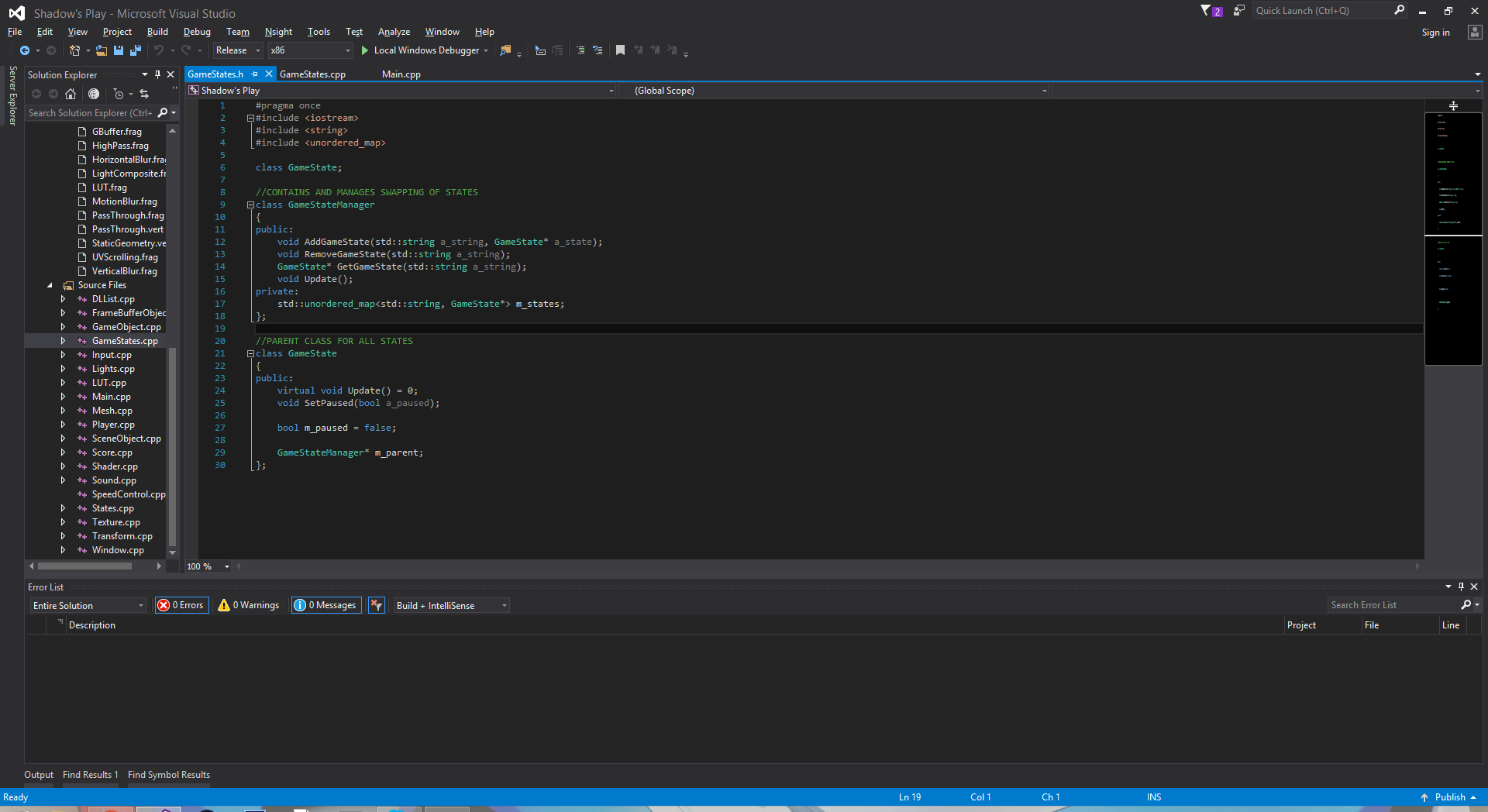


Figure 4: Game State Manager class in the header file

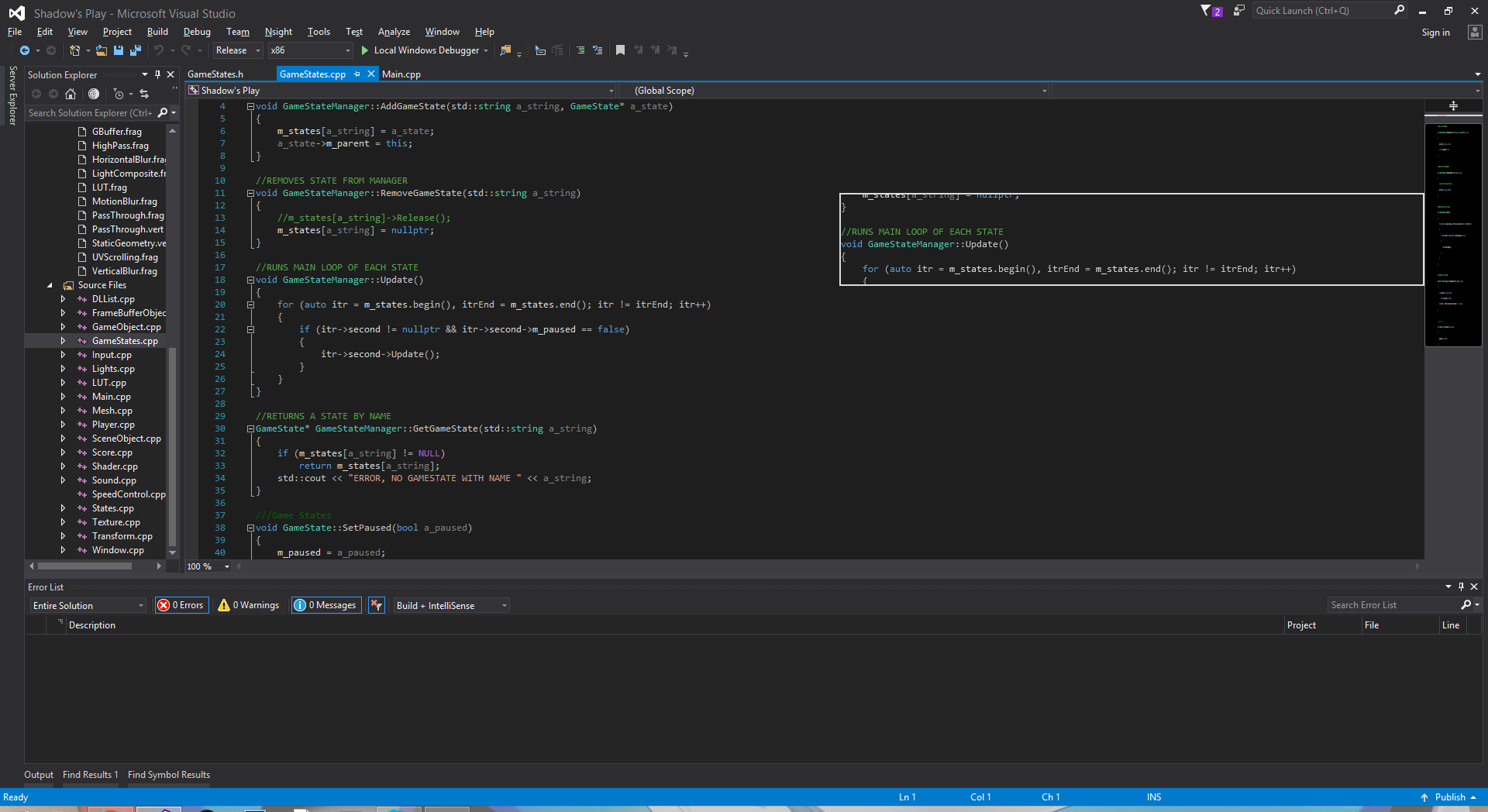


Figure 5: Game State Manager in the source file

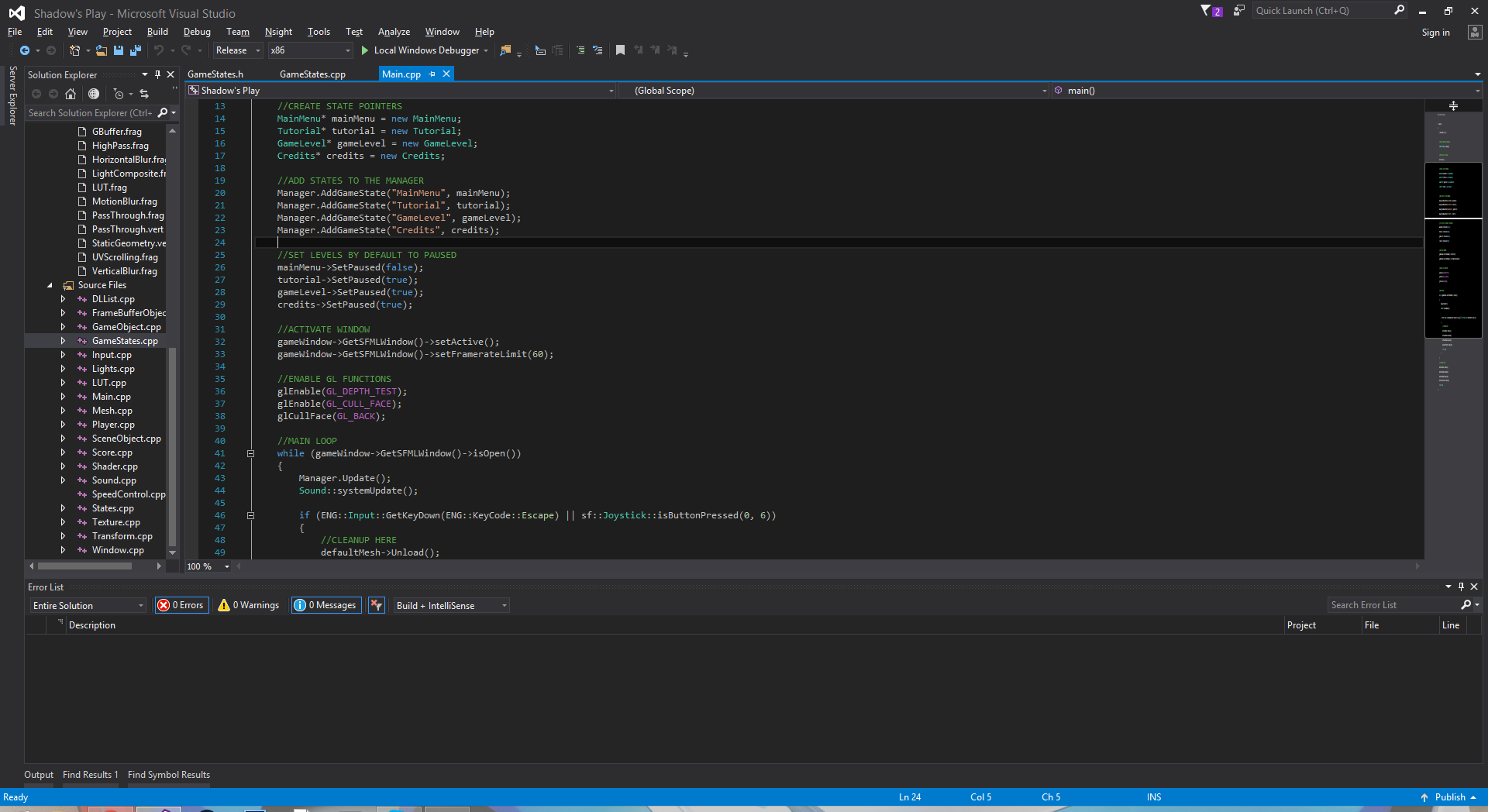


Figure 6: Creating the Game State Manager in Main.cpp

**Inheritance:**

Each of our game state is inherited from the parent game state class in States.h. We have a Game Object Class that is the parent class of our Player class & Scene Object class.

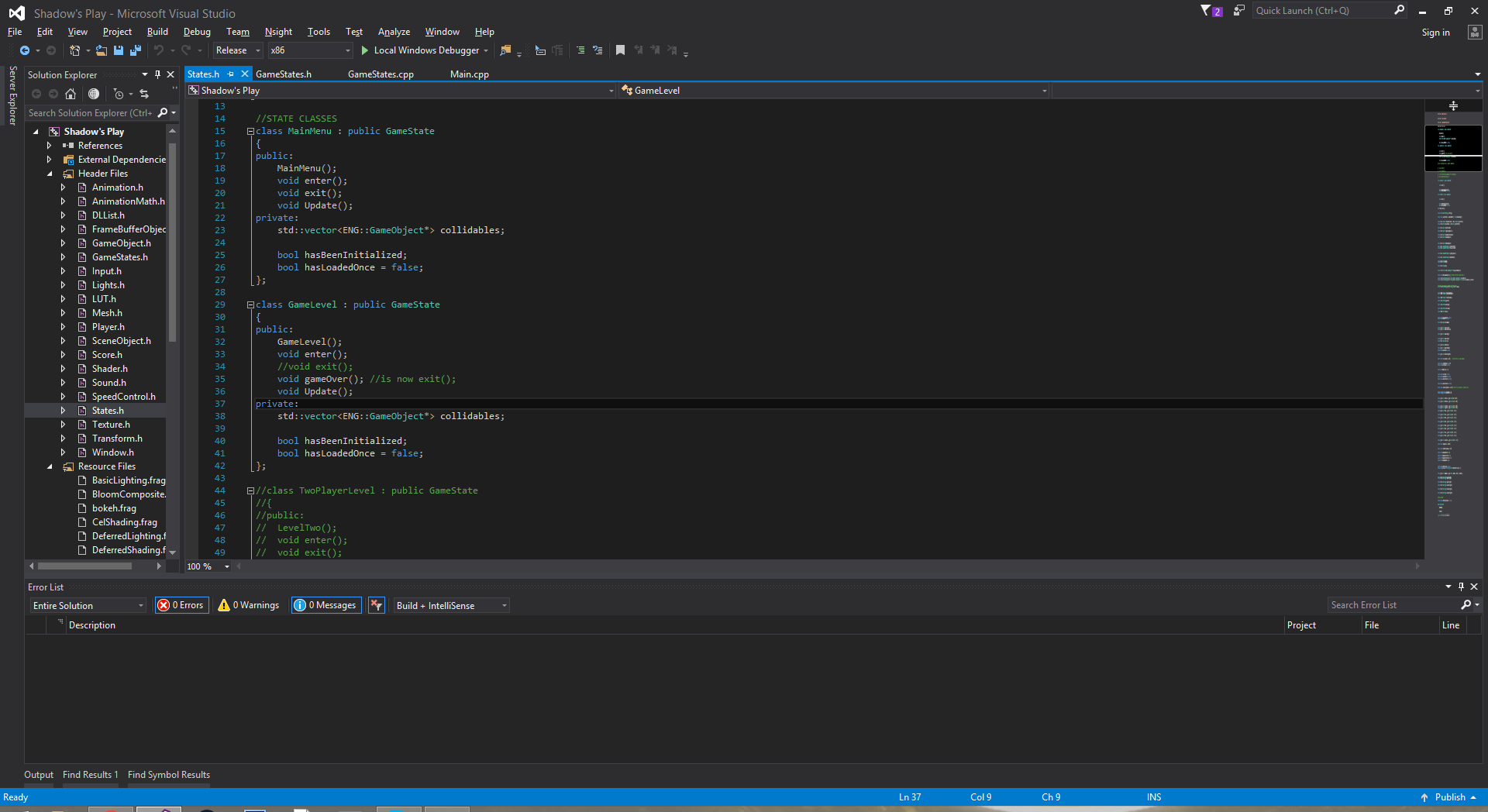
****

Figure 7: Children of the GameState Class located in States.h

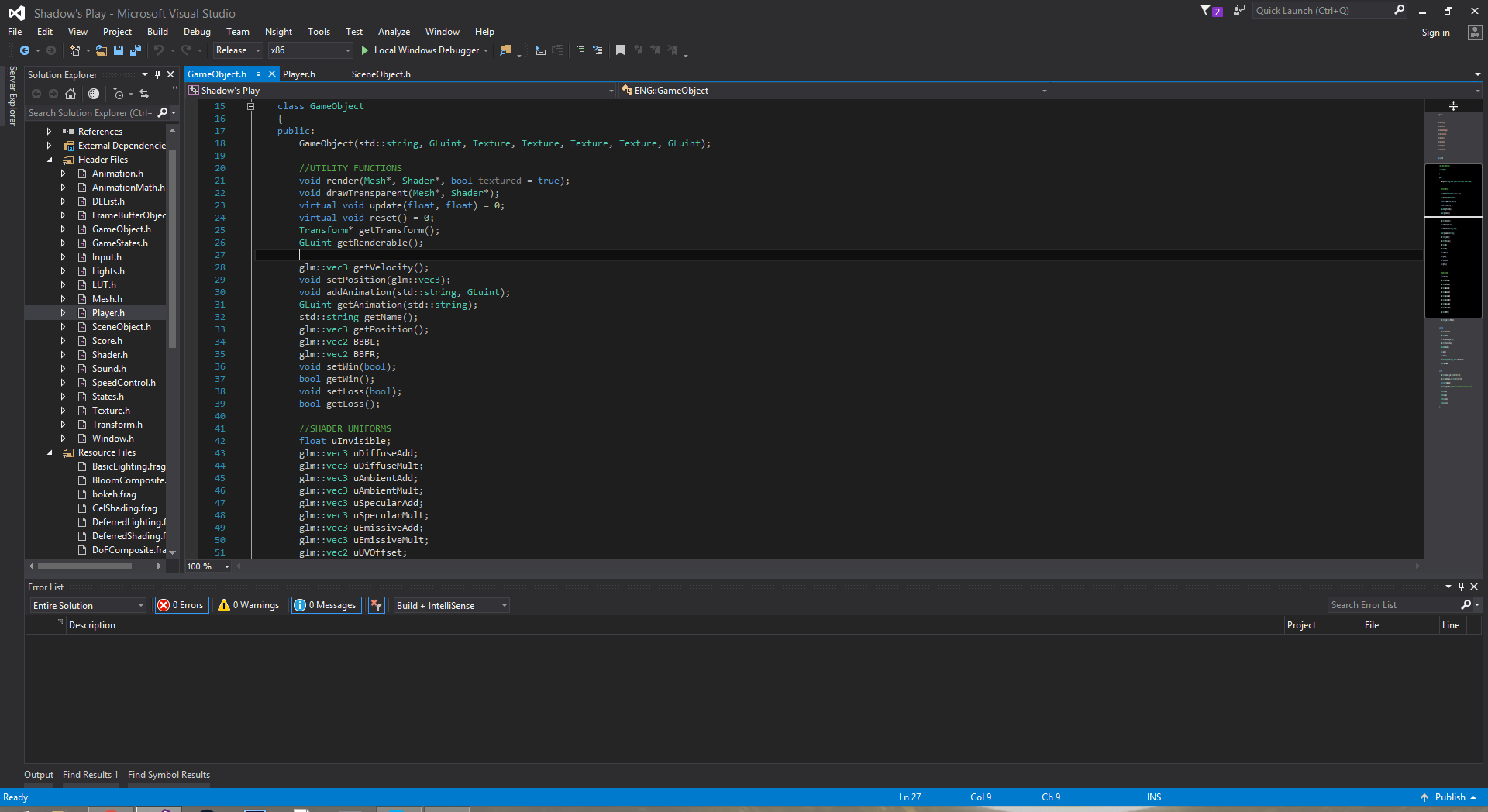


Figure 8: Parent Class of Player and SceneObject

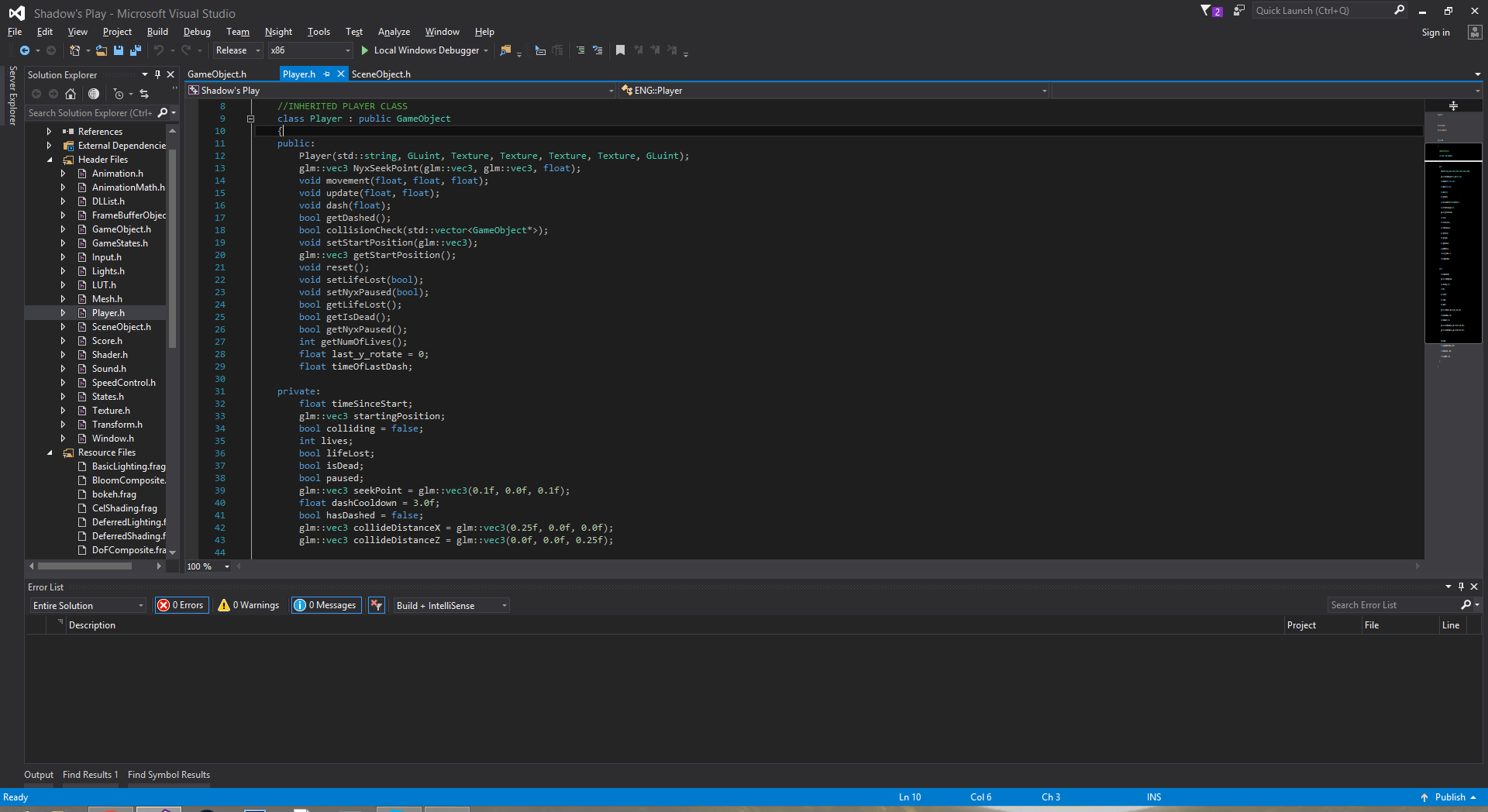


Figure 9: Child Class Player

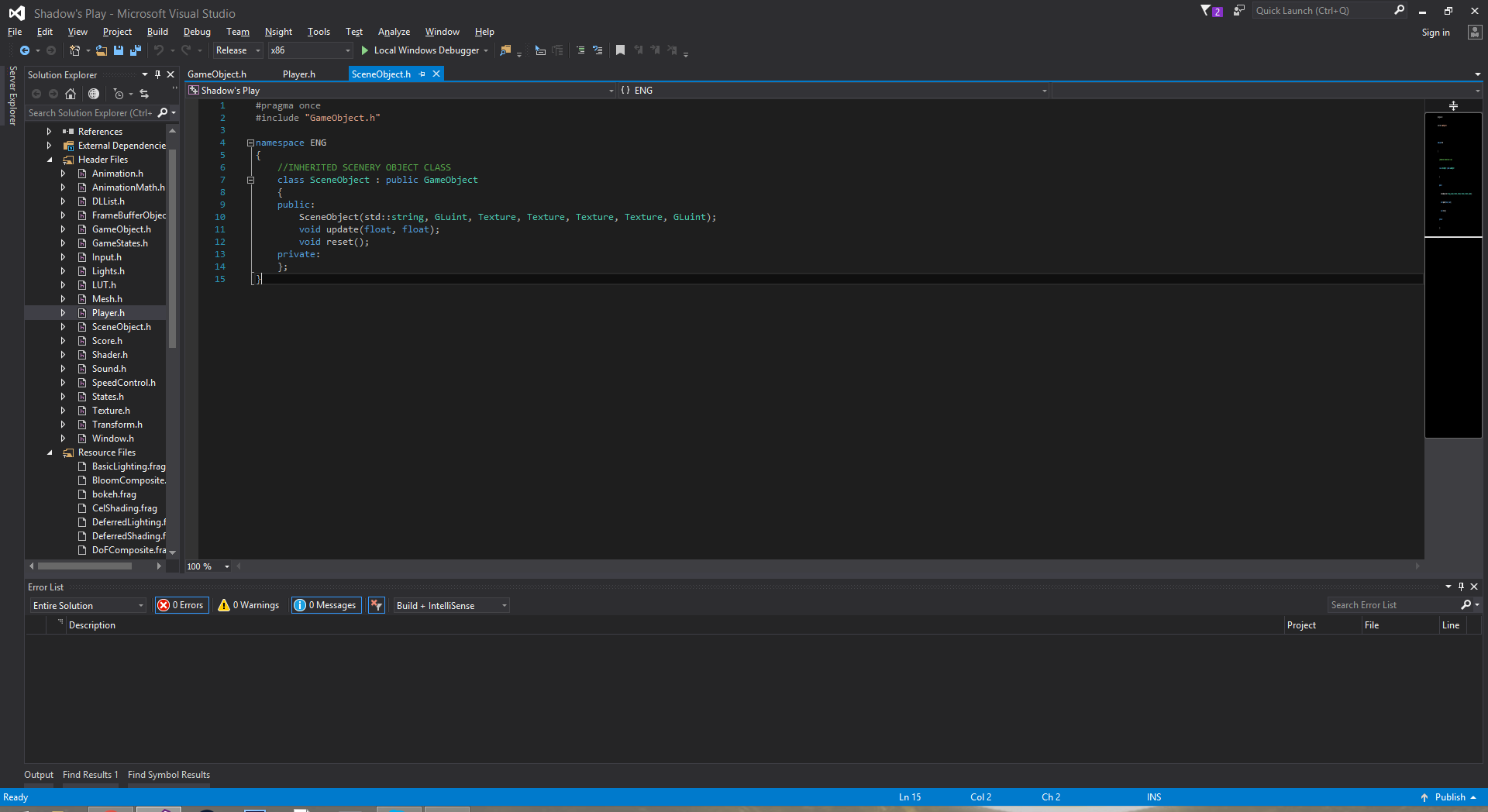


Figure 10: Child Class SceneObject

**Sorted Additional Data Structure:**

We implemented a sorted doubly-linked list similar to one used in assignment 3, but it was adapted to store scene objects to be drawn in an orthographic view with a priority given by an artist or a designer. This priority determines the layer on which the scene object will be drawn.

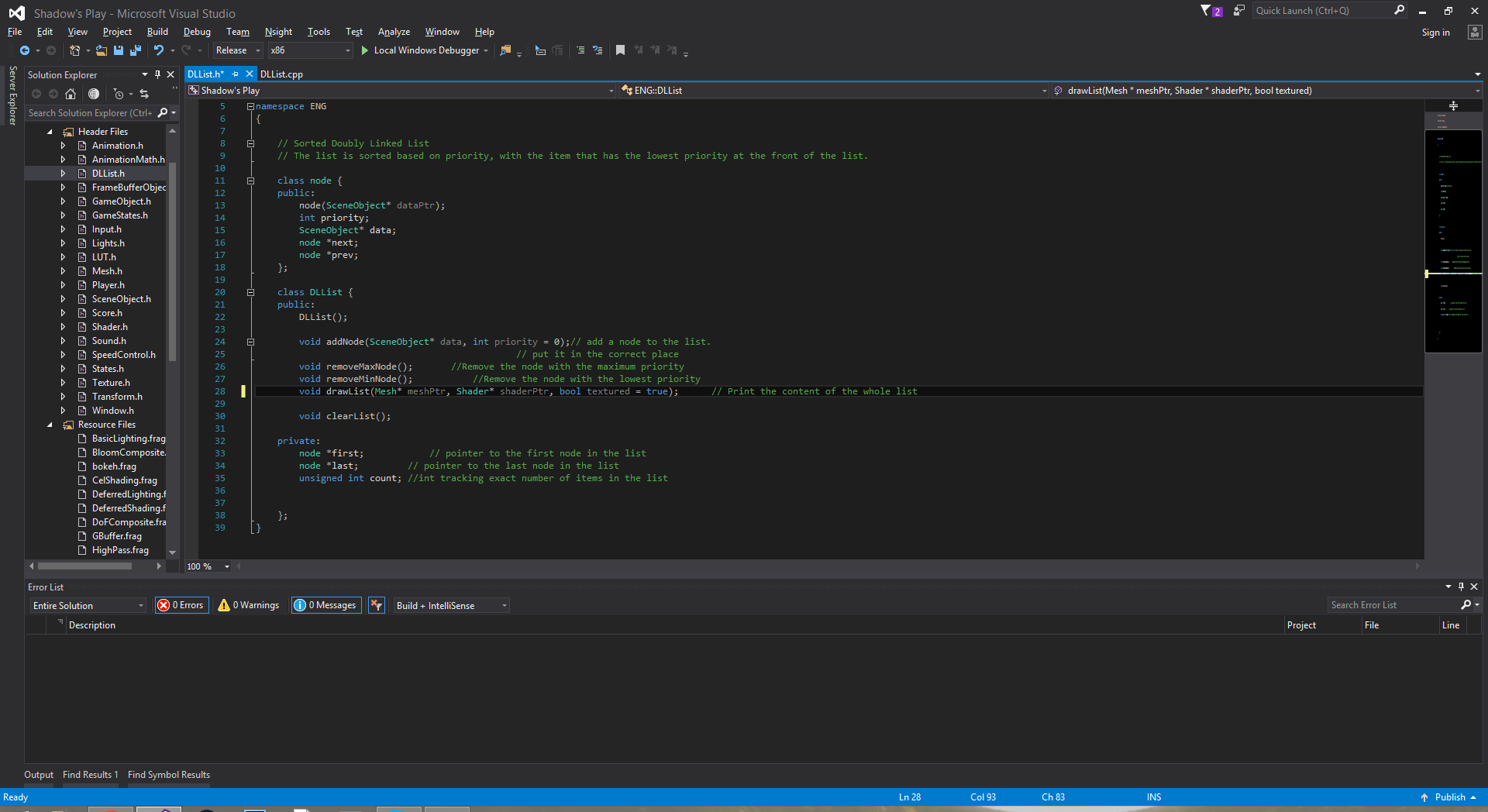
****

Figure 11: Dynamic Doubly-Linked List Queue

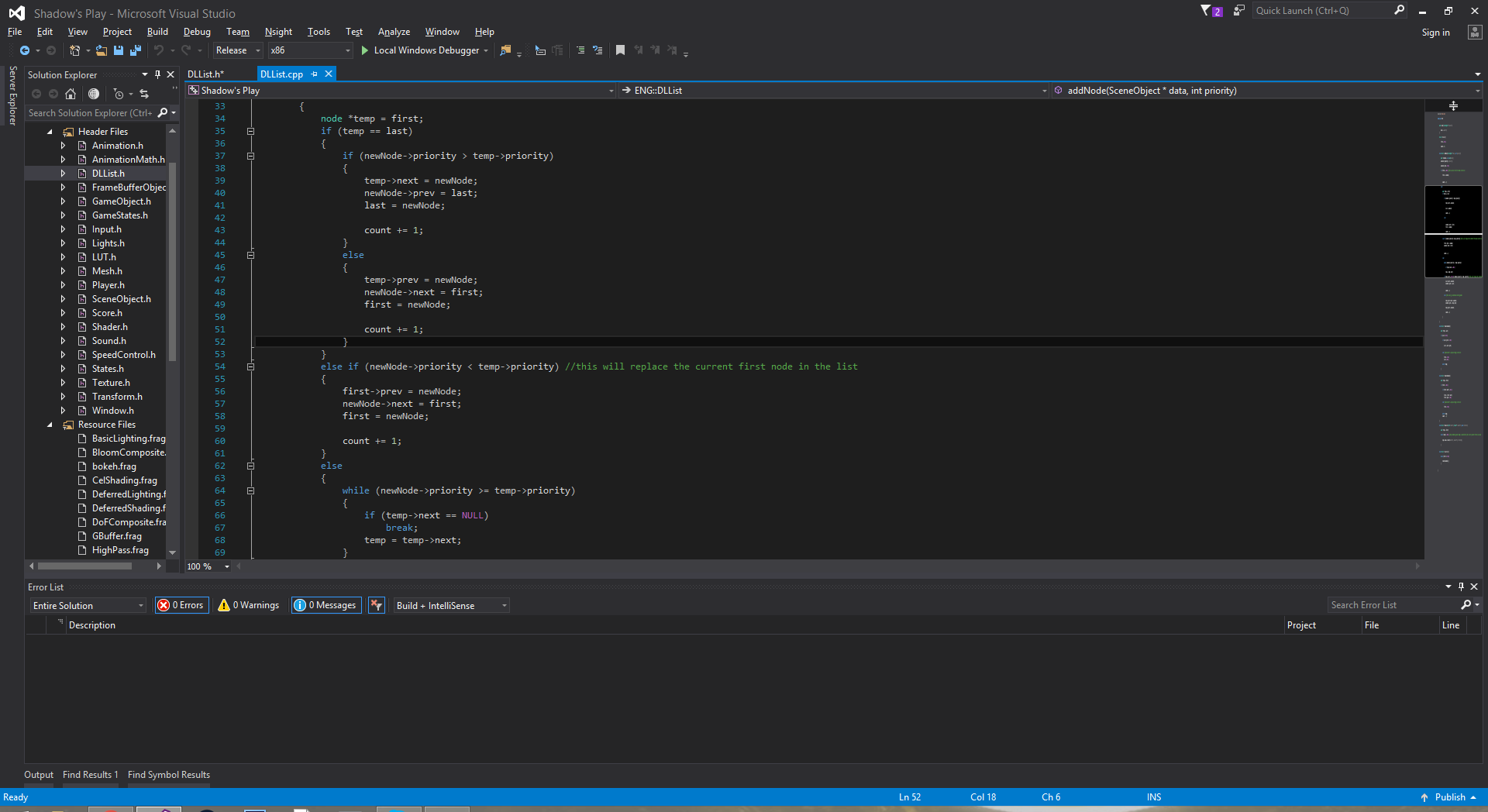


Figure 12: Code Snippet of the sorting algorithm used in the insertion function for the list

**Exception Handling:**

We handle exceptions in the loading process of all the shaders, textures, sounds, etc.

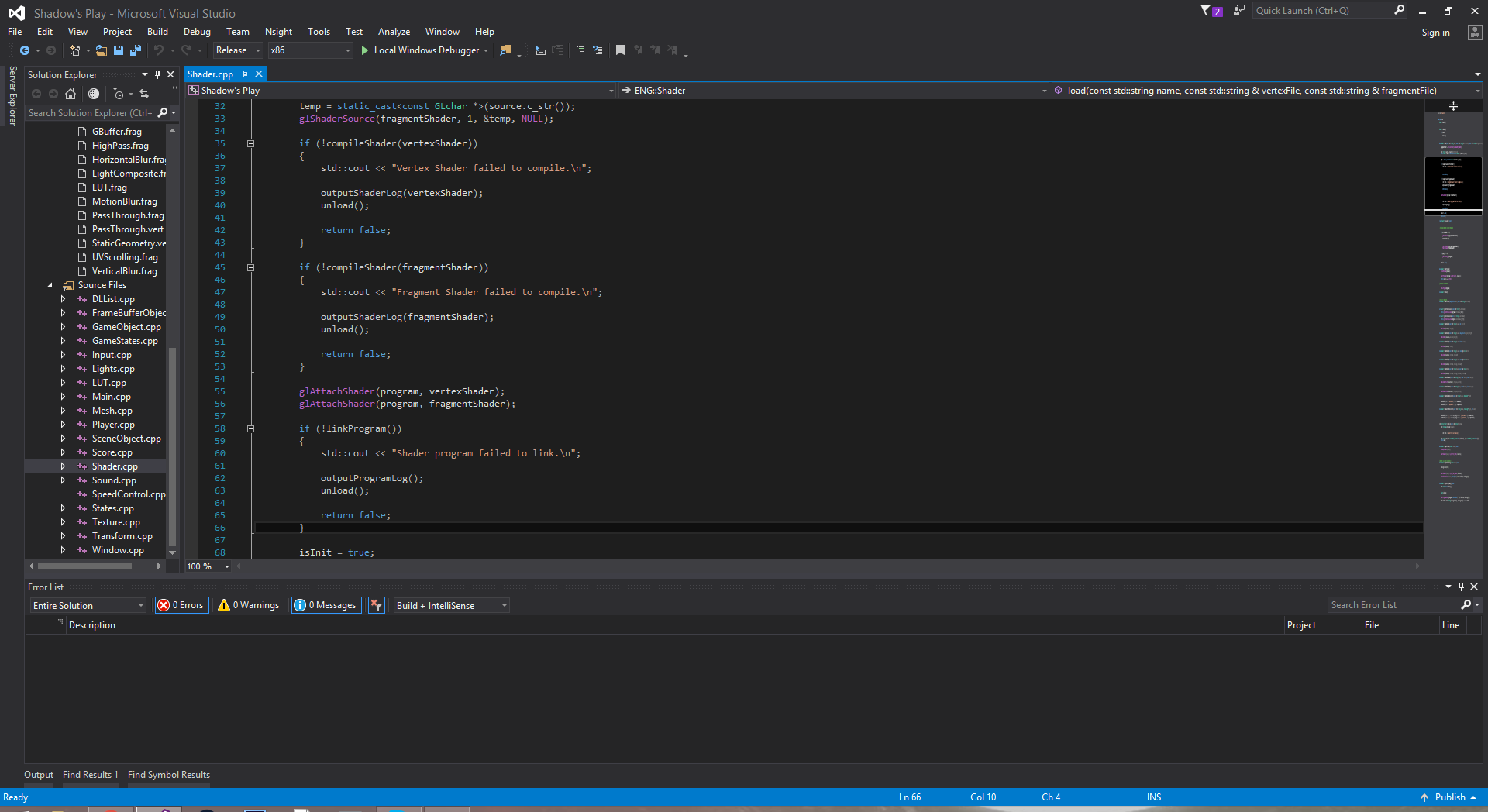
****

Figure 13: Implementation of OpenGL exception handling when loading shader programs

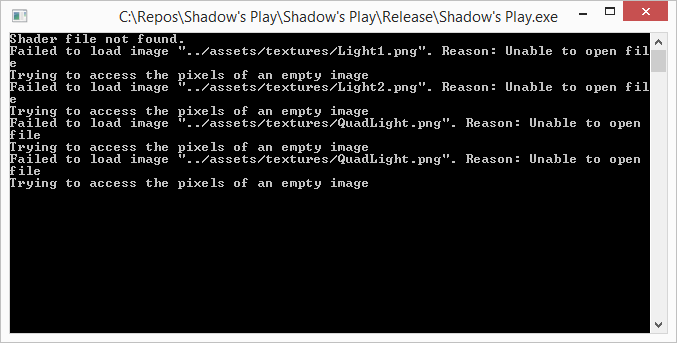


Figure 14: Console Output featuring one shader exception and several texture exceptions

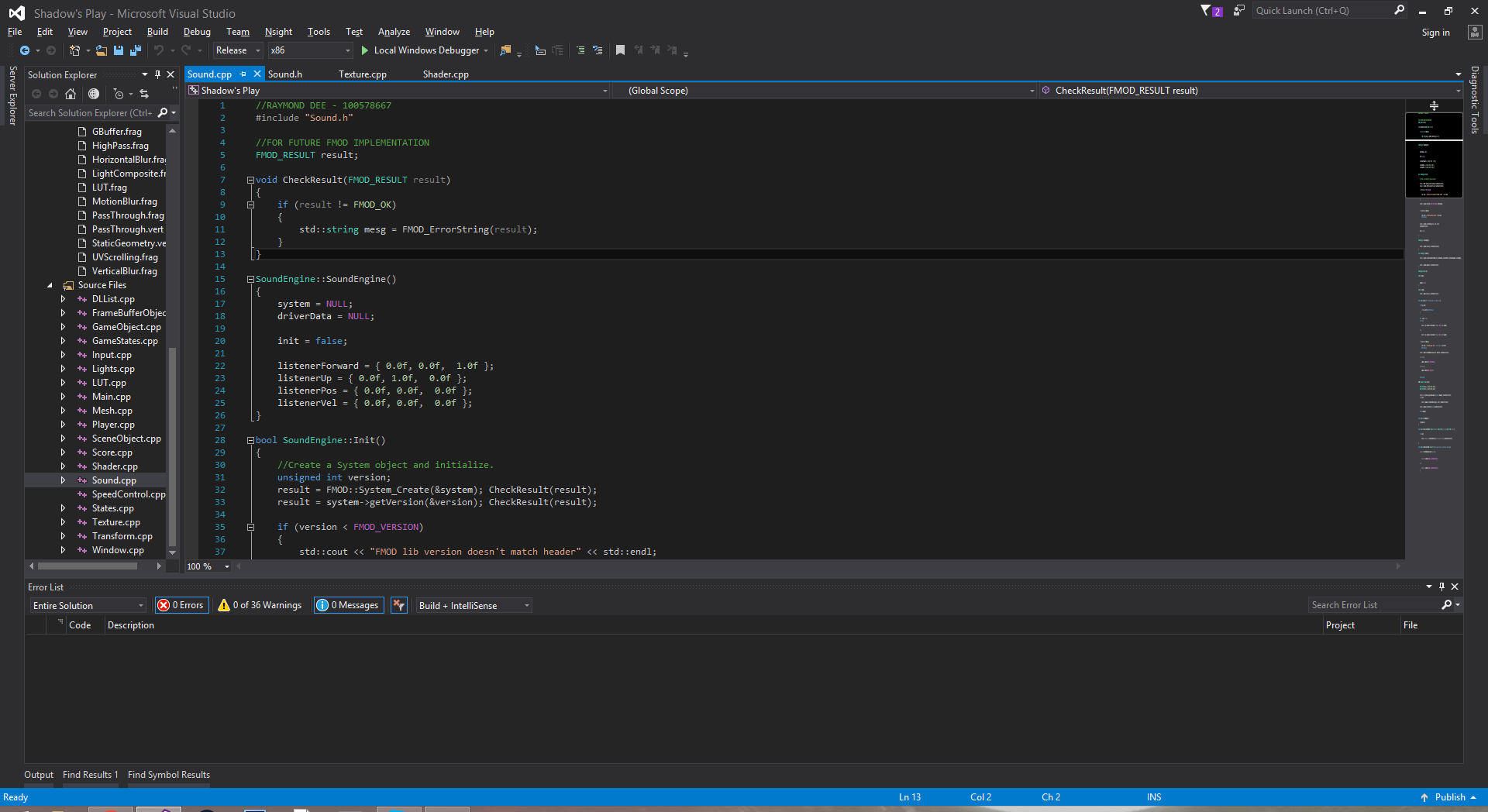


Figure 15: Exception handling of FMOD Sound systems