**PLATFORM UESED:**

I have not used any other third-party libs other than provided libs and engine.

* Language: C++ 17
* Compiler: VS 2019

**ARCHITECTURE:**

Classes-

* CandyCrushGame: This is main class for the game, it manages whole game system. It is derived from engines game class and overridden its methods like update, render etc...
* Gameplay: This class has all gameplay related code like
  + Managing the mainboard
  + Handling inputs
  + Game logic on Score and Timer update and its related checks
  + Resetting whole system for the new game
  + change of State and menu request as per gameplay actions
* MenuManager: This is responsible to render, update all the menus inside the game. It holds object of Menu class and takes care of updating and rendering current menu.
* Menus: Menu class is base class of following in game menus:
  + - Start Menu
    - Game Over Menu
    - Setting Menu
    - Help Menu
    - Profile Menu

Every menu has its UI and logic components with them which gets updated and rendered from Menu Manager.

* StateMananger: State Manager Manages game state:
  + - Init State
    - Menu State
    - InGame State
    - Exit State

This class notifies the listeners if any state change. Default state is init state

* **InputController**: This Class takes care of all types of input actions and signals to mainboard and gameplay about input change. Currently only mouse input is supported.
* **MainBoard**: It contains the game board (8 X 8 ) and the associated functionalities like checking a valid move, finding horizontal/vertical matches, updating grid if a match occurs. It also swaps if move is valid
* **Time Manager**: Game is Timer Based; we have defined a game session for 60 sec. This class tells remaining time and helps to understand if game is over.
* **CandyFactory**: This Produces candies. With the help of random function we generate random candy.
* **Candies** – We have 5 different cadies each one is described as a class.
* **Isender**: This is a subscriber base class/ Interface for state manager to broadcast state change:
* **Ilistener**: This is observer base class/Interface for Candy Crush game class so that it can listen for any state change. When there is state change, “OnStateChanage” gets called. CandyCrush Game have defined state machine logic on the basic of state change
* **Singleton**: This is templated singleton class which return unique ptr of T type reference when we call GetInstance() method. I have used CRTP pattern to use this class

Design Patterns Used-

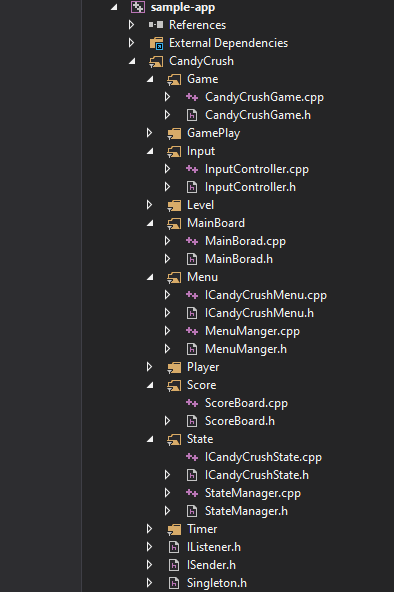
* Singleton
* Observer
* Factory

Modern C++ Idioms Used-

* STL array, vector
* Virtual, overriding
* Smart pointers..

Coding Standards-

* Uniform Naming style for function, variables and classes.
* Tried to keep code as readable as possible. Improved readability of the code by consistent naming style and short comments.
* Related data is Encapsulated with appropriate level of abstraction.
* All classes are easily extensible for future scope.



* Each class resides in a separate file.
* I tried to keep use of application memory as low as possible by passing function parameters by reference, saved object copies
* Tried to follow good programming ethics and OOPs principles
* While designing the architecture of the application I consider SOLID principles
* Code with appropriate comments

**HOW TO PLAY:**

Simply click on exe file.

**FUTURE ENHANCEMENT:**

* multithreading will improve it.
* dynamic board (input dimensions from the user)
* use of constexpr to build maximum code at compile time/less run time overhead
* simulate the motion of the candies like (falling of the candies) with the help of animation and sound.