Report: Project Meat Boy

This year's project consists of designing and implementing an interactive game, inspired by Meat Boy, in C++ using the SFML graphics library.

Before explaining any of the features or design choices I will first explain how the core game works. Upon first launching the game, you will be presented with a menu containing a list of levels to select from. Upon selecting a level, you spawn inside a building made of walls where to goal is to reach as high as possible while jumping against walls until you hit the goal. Upon succeeding, you enter the next level. In each level there are 3 main entities: player (you), goal and walls.

Logic (static lib):

Abstract Factory:
The abstract factory, together with its derived class Concrete factory in the game representation, is a class which is responsible for creating the entities. The Abstract factory pattern (together with the Concrete factory) is used with this class.

Subject and Observer:
Subject is an abstract class containing a map of event and list of pointers to observers pairs (to know which observers are registered for 1 type of event of the subject). The Subject class (with its derived classes), the Observer class (with its derived classes) and the World class are implemented using the MVC design pattern. Here the Subject is the Model; the Observer is the view and the World is the Controller. The Subject class and the Observer class together are implemented using the Observer pattern.

EntityModel:
Entity Model is a derived class from Subject. The class contains common functions that are used by all the classes that are derived from this class to avoid duplicate code.

Player:
The Player class is a class derived from Entity Model. In this class I handle all the movement of the player and collisions between the player and other entities in the game. The reason behind handling collisions in the Player class is because i use the calculated distance components (to add to the player position) with the logic for detecting collisions.

Wall:
The Wall class is a class derived from Entity Model. it describes all the features and functions all sorts of walls in the game have in common.

Camera:
The Camera class is used to project logical coordinates (coordinate system) to pixel coordinates (for renderwindow). It has 2 bounds: upperbound and lowerbound that specify the height of the camera in the coordinate system.

GirlGoal:
The concrete goal class of the more general Goal class. Gets the name of the texture to be used with the entity as input. Its the most basic form of a goal.

MeatBoy:
The concrete player class of the more general Player class. Gets the name of the texture to be used with the entity as input. Its the most basic form of a player.

SimpleWall:
The concrete wall class of the more general Wall class. Gets the name of the texture to be used with the entity as input. Its the most basic form of a wall.

Stopwatch:
Used to keep the game running on the same speed on different machines with different speed. Keeps calculating dt which is the time that a particular frame took to complete its calculations and renders and sleeps if its less than 1000/(Frame Rate cap).

Game representation:

kNow that we have seen the logic of the game it is important to know how we are going to display everything. Displaying everything and all the textures and sprites operations happen in the game representation part.

ConcreteFactory:
The Concrete Factory class is the class derived from the abstract factory class. In this class we do the same as in the Abstract factory class

EntityView:
The EntityView class is derived from the Observer class. This class is used to draw the sprites of the different entities.

Game:
The Game class is the class where we handle everything with displaying and input handling and running the main game loop.