**Project - Bomberman**

This program is a rebuild of the classic Bomberman game.

The goal of the game is to gain as many points possible and move through the levels until the game is over.

The player can select a level through the menu using the mouse or the keyboard.

The player moves around using the arrow keys and lays bombs with the B key. To restart the current level the user can press the R key, and in order to quit the game the player can press the Q key.

The player is limited by time, and number of lives.

**Design**

**The Classes created for this program:**

* GameObject – Holds the very basic data that any object in the game would need, that’s an abstract class.
* StaticObject – Inherits from GameObject, holds the next level of functionality for the static objects of the game, that’s an abstract class.
* DynamicObject - Inherits from GameObject, holds the next level of functionality for the dynamic objects of the game.
* Player – Inherits from DynamicObject, holds the methods and functionality of the player.
* Guard- Inherits from DynamicObject, holds the functionality needed for the guards in the game.
* StupidGuard – Inherits from Guard, holds the functionality for the stupid guards that move randomly.
* SmartGuard – Inherits from Guard, holds the functionality for the smart guards in the game that chase the player.
* Wall – inherits from StaticObject, holds the functionality for the Walls.
* Box - inherits from StaticObject, holds the functionality for the Boxs.
* Door - inherits from StaticObject, holds the functionality for the Doors.
* Explosion - inherits from StaticObject, holds the functionality for the Explosions
* Gift – Inherits from StaticObject, holds the functionality of the three different gifts in the game.
* Bomb – Inherits from StaticObject, holds the functionality of the bombs in the game.
* Controller – Holds the functionality of the entire game and connects all of the classes.
* Graphics – Holds the graphic parts of the game and loads the correct textures to the places they are needed.
* Level – Holds the data and functionality of the level.
* Screens – Manage the side screens of the game. This class is the only who contain its own textures. (no involved Graphics.)

**Files created:**

GameObject.h – Holds the GameObject Virtual data and meth/ods (Abstract class).

StaticObject.h – Holds the StaticObject Virtual data and methods (Abstract Class).

DynamicObject.cpp/h – Holds the DynamicObject class data and methods.

Player.cpp/h – Holds the Player class data and methods.

Guard.cpp/h – Holds the Guard class data and methods.

StupidGuard.cpp/h – Holds the StupidGuard class data and methods.

SmartGuard.cpp/h – Holds the SmartGuard class data and methods.

Gift.cpp/h – Holds the Gift class data and methods.

Bomb.cpp/h – Holds the Bomb class data and methods.

Controller.cpp/h – Holds the Controller class data and methods.

Graphics.cpp/h – Holds the Graphics class data and methods.

Level.cpp/h – Holds the Level class data and methods.

Wall.cpp/h – Holds the Wall class data and methods.

Box.cpp/h – Holds the Box class data and methods.

Door.cpp/h – Holds the Door class data and methods.

Explosion – Holds the Explosion class data and methods.

Screens.cpp/h - Holds the screens class data and methods.

**Levels format**

The levels are loaded from a text file called Levels.txt.

The first line of each level contains the width, height, number of bombs and time limit in seconds.

After that the level in this format: #-wall, @-box, !-guard, /-player, +-gift, D-door.

The levels are separated by one blank line.

**Data structures**

-Vectors: These data structures hold all of the data we use- maps, Guards, Bombs and gifts.

-UniquePtr: In the program we have a vector with this type of pointers that hold the two kinds of guards we have and all rest staticObjects (Walls, Boxes, door..) And for Bombs as well, separated uniquePtr.

-We are using double dispatch to handle collision within all of the GameObjects

**Known bugs**

-