# Bubble Struggle Game

**Presenters:** Efrayim Sztokman ID 31570413 and Itay Mizrahi ID: 305360695

# General Explanation:

This Is a game Where the player/s must avoid bouncing balls , that bounce all over the screen the player can shoot the balls only once all of the balls that are on the screen are gone can the player/s move on to the next level.

# General Design:

A command Design pattern is used to determine which type of game is played.

The controller loads a level from the file level.txt onto the board via a factory class.

The objects ae held in the board.

The controller runs the game by timing when the objects get updated, collide and when they get printed.

The Controller ends the game or loads another level (or the same level again) if necessary.

The updates and collisions take place in the board timed by the Controller.

# File list:

**Menu**- in charge of sending the user to the specified location of the game:

The user may choose from

1) Single player – where one player plays alone

2) Multiplayer—where two players play together

3) Exiting the game

4) Seeing the keys needed to be pressed.

**Board**- A singleton class in charge of holding the game objects updating and drawing them.

**Controller** - The board's ‘brain’. He is responsible synchronizing objects movement and levels being loaded onto the board.

**GameToolbar** - Displays information of the game –lives and time left.

**ResourceManager**-A singleton class that holds all sounds and textures in a map stricter for easy access of the other classes

**LevelLoader-**holds a two dimensional array of strings each representing a level that can be loaded onto the board with DrawableFactory.

**Drawable**- a base class for all of the object that can be drawn by the board onto the screen.

**Moveable**- derived from Drawable a base class for any moving object

**Unmovable**- derived from Drawable a base class for any static object

**AbsBall—**abstract base derived from Moveable. A class for all of the different types of balls from the size 10, 20... 80 radiuses

**Player—**base class Derived from Moveable .for different types of players

**AbstractWeapon-** an abstract base class derived from Moveable .for all different types of weapons in the game.

**AbstractItem—**an abstract base class for all of the different items in the game

**Derived from AbsBall:**

**1) Ball-** The most basic ball bounces relatively normally, when hit splits into to smaller balls.

**2) UpdownBall –** bounces up and down on until hit then splits into 2 balls smaller balls of class Ball

**3) DirectHitBall—**a ball the size of 10 radius that can only be removed if hit directly.

**4) ZeroGravityBall—**a ball with no gravity on it that acts just like Ball otherwise.

**5) BossBall-** a hovering ball that shoots projectile balls (described below) at the players of the game, when hit hovers closer to the player until hit 5 times then removed

**6) ProjectileBall—**a ball the size of 20 radius that is shot by the BossBall at the player

When hitting another object splits into two balls of Ball class.

**7) HeldBall-** a template class used by Door (described below) acts like the ball that is in its template, holds an extra pointer for telling when the last ball created by the ball splitting has been removed, for the door to know when to open.

**Derived from AbstractWeapom:**

**1) SimpleArrow-**the default arrow, moves up until collision with Wall (described below) or AbsBall derived class.

**2) FrozenArrow-**acts like Simple Arrow except stays put when hitting a Wall

**3) LaserWeapon- is** a projectile that gets bigger when moving.

**Shield-**Derived from Moveable- surrounds the player that owns it for

A brief few seconds, all of the AbsBall Derived object bounce off of this protecting the player.

**Derived from AbstractItem:**

**1) SheildItem—**gives the player a temporary shied that all balls bounce off of.

**2) LaserItem-** gives the player a LaserWeapon

**3) FrozenItem-**gives the player a FrozenArrow Weapon

**4) SimpleArrowItem-**gives the player a simple arrow

**4) CashItem-**adds score to the players

**5) ClockItem-**adds time to the level

**6) OneUpItem-**gives the player another life.

**Derived from Player:**

**1) PlayerOne-**the basic red player

**2) PlayerTwo—**a blue player added only in multiplayer.

**DrawableFactory-** factory for creating all different type of Drawable derived objects onto the board. Created by a string. Holds a map that maps a string to a creation function pointer

**DrawableFactoryFunctionTemplates-** namespace for template functions that can be used as function pointers for the DrawableFactory.

**Wall-** Derived from Unmovable this class is an unmovable wall in the game that neither players nor balls can pass through.

**Door** Derived from Unmovable this is a wall that holds a ball either by pointer or by Heldball wrapper that is removed only when either the HeldBall is removed or the ball itself

If held without HeldBall.

# Main data structures and their roles:

- A map of textures in the ResourceManager, contains all the required textures in the game.

-A map of sounds in ResourceManager contains all of the sounds for the game.

A vector of players in Board

2 vectors one of Moveable and the other of Unmovable for the game objects in the board.

A two dimensional vector of strings each representing a level in LevelLoader.

A map of function pointers in DrawableFactory – for creating object onto the board.

**Known bugs**:

# Design Patterns that are noteworthy:

**Double Dispatch—**for collisions

**Singleton—**for both the board and ResourceManager

**Factory –** used for generating the game objects.

**Command –** used for the menu

Other notes: