**Sw Devt IV – Advanced .NET 420-411-DW  
Project 1 – Tetris  
Phase 3 - MonoGame**

**Project Phase 3:** Add a Monogame UI which will take care of the game loop, user input and drawing on the screen.

You will add a MonoGame Windows Project project to your solution. Within the game project, you will build the classes similar to that shown on the next page. You will notice that the MonoGame classes use the business classes extensively. The ShapeSprite class is responsible for rendering the active shape, as well as collecting keyboard input. The BoardSprite is responsible for drawing the Board (the pile is distinguished from the rest by its Colors). The ScoreSprite is responsible for writing score-related information, such as the level, number of lines cleared, and the score. The Game1 class instantiates the appropriate business classes and the sprites, and adds the sprites to its components list. It also observes the Board class to know when the game is over: at that stage it will remove the ShapeSprite from the components so that it is out of the game loop, and informs the ScoreSprite so that a message is displayed.

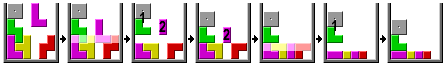
I have provided you two block png: one to use for empty Board cells, and one for filled cells / Blocks. Use the SpriteBatch.Draw overload to tint the filled blocks with the correct colour.

Note that the ShapeSprite is responsible for asking the current piece to move down. The frequency with which this happens is based on the level:

dropDelay = (11-level) \* 0.05 seconds

In other words, at level 1, the shape will drop at every ½ second, increasing at every level by 0.05 seconds, or 1/20 of a second. Since the maximum level is 10, the quickest that the shape will move down is every 1/20 of a second. The game loop iterates at every 1/60 of a second, so you will need to use a threshold in your Update method to throttle how often you ask the shape to move down.

Refer to the sequence diagrams below to understand how the classes interact with each other.

**[](http://upload.wikimedia.org/wikipedia/commons/1/1b/Tetris_gravity_(natural).png)Improving your project!**

Satisfy the over-achiever lurking within you! Here are some optional suggestions for you to tackle only once you get the basics working:

1. implement a more intelligent pile-clearing algorithm. Image above from [Wikipedia](https://en.wikipedia.org/wiki/Tetris) to explain. Instead of simply deleting a filled row, and descending all rows above, implement a flood fill algorithm. In other words, don't stop at the 4th frame above.
2. show the next shape
3. allow the player to hold a piece for later use
4. enable clockwise rotation
5. show "ghost" shapes that indicate where the shape would fall if there was no further manipulation
6. Allow the player to pause/resume the game at any time.
7. Add music!
8. Save and read the high score from a file. Save one or multiple high scores, with the players' gamer tags.

**Updated Tetris Business classes**

C:\Users\jnilakantan\AppData\Local\Microsoft\Windows\INetCache\Content.Word\tetris.png

**Tetris Monogame classes**



**Sequence Diagram - Instantiation**

C:\Users\jnilakantan\AppData\Local\Microsoft\Windows\INetCache\Content.Word\TetrisInstantiationSequenceDiagram.png

Notes:

* in my design, the Board business object is created first, and it instantiate the ShapeProxy, passing it a reference to itself (i.e., this)
* the ShapeProxy loops through and creates all the concrete Shapes. In my design, I just create the 7 shapes and keep reusing them by resetting them
* each shape creates the 4 Blocks that it contains
* the Score business class also needs a handle to the Board, but only to subscribe to the LinesCleared event; it doesn't need to keep it as an attribute
* All sprites need a handle to the Game1 object
* The ShapeSprite takes care of drawing the current shape, which is wrapped by the ShapeProxy. The IBoard has a Shape property which is used only for the ShapeSprite constructor to get a handle to the IShape
* Finally, if you want a border around the grid when you are drawing it (e.g., x pixels, or x widths/heights of the block images), make sure you tell both the BoardSprite and the ShapeSprite so that they can add this to their Draw calculations (i.e., you may want to pass additional values to the sprites in their constructors).

**Sequence Diagram - Update, and hit pile (game not over)**



Notes:

* the ShapeSprite will ask its IShape (ShapeProxy)to MoveDown every time its counter reaches the threshold, as determined by the level. Update is called every 1/60 second, so make sure you throttle before asking the shape to MoveDown
* The ShapeProxy will relay the method to the current shape: the ShapeProxy is acting as a wrapper, and will just relay method calls
* The concrete Shape will loop through its Blocks, to check if the move is valid: each Block will check if they can move down by 1 (i.e., is the Board unoccupied at position (x, y+1) and not at bottom)
* If all the Blocks can do the move, the Shape will ask them all to do it
* If any Block cannot move down, the Shape raises the JoinPile event
* The ShapeProxy catches the event, and turns around and fires its own JoinPile event: this means that the Board only ever needs to subscribe to the ShapeProxy's event, and all the events of the concrete shapes are subscribed to by the ShapeProxy
* The JoinPile event is caught by the Board. This will cause if to go through the Blocks of the IShape (using the indexer) and changing its internal array to be the colour's of the Shape. It then checks for lines, and if it clears any, it throws a LinesCleared event. It will also check if the game is over. Finally, the Board will ask the ShapeProxy to deploy a new shape