Requirements Document

In order to make this CS-401 project a success, we need to list our requirements. Please also list any goals That you have. This will be our pre requirements document. I’ll use it to build the final one that will be turned in on 6/5.

Project Overview:

The aim of the project is to apply the process of Software Development to a game design. To this end, we will create the classic match 3 game Collapse! The goal of the game is to clear as many blocks as possible in the time allowed. If the player matches 3 or more bricks that are touching, the blocks disappear and the player is awarded points. The larger the chain, the higher the number of points awarded. The player wins if the timer expires. The player loses if blocks reach the top of the screen. The game goes is an arcade style endless game. The blocks get faster in later levels.

The aim of the game is to appeal to casual players. The game should be playable on as many devices as possible. This will help the game appeal to the highest number of players. The game should feature a simple input to help this appeal. (The only user input is a mouse click or touch) The game will be easy to play, difficult to master. The higher the chain reaction, the more points are awarded. For example, the player can match 3 of the same color blocks for a base amount of points. On the other hand, the player can wait or the board to fill up , and match more blocks for a higher score.

**Target Platforms:**

Web browser on Desktops

Mobile Web browsers

The target platform for the game is a desktop web browser. Since the goal of the class is to produce an open source project, it makes sense for the software itself to be accessible on multiple platforms. The secondary target are mobile device browsers. If we use an HTML 5 based game engine, we should be able to support many platforms with one build.

Link to Game Design Document: <https://docs.google.com/document/d/1_Pt5zu71iiOrx-OTe1wfYGwxJY30v_Rgj6jvEWTwE-0/edit>

**Project Deadline: 7/24**

**8 Weeks total time**

Goals:

* To recreate the Software Development Process in the context of the classroom
* To infuse real world features into a game (Databases and Facebook Integration?)

Functional:

* Collision Detection
  + Block detects when finger gesture or mouse clicks on it
  + Block detects when it is touching 3 or more objects
  + Block detects when it is in contact with the top of the Screen
* Basic Game Manager
  + Level Loader
    - Load the current level on initiation
    - Handles Timer
    - Handle the end of level when timer ends
  + Row Generator
    - Generate the 4 colored blocks in random patterns
  + Score Keeper
    - Award player points for tapping 3 or more matching blocks
    - Multiply score if more than 3 blocks matching
  + Win Condition
    - Calculate end of Level Score
    - Display Score in UI
    - Load Next Level
    - Make blocks appear faster
  + Loss Condition
    - Calculate Final Score
    - Display Final Score
    - Handle game entity Cleanup
    - Display Play Again UI
    - Restart or quit Game
  + Handle what happens when player hits pause button
* User Interface
  + Main Screen
  + Display End of Level Win Screen
  + Display Loss Screen
  + Pause Menu Display
* Sprite Manager/ Playfield Generator
  + System to handle particle effects
  + System to handle drawing of sprites on Screen

Non Functional:

* Need a website to Host the game
  + Website should hold database of high scores
  + Game Should use database to keep score
  + Game should have online leader boards
* Game Must have Sound
  + Gather sound from open source places
* Game should have social features
  + The game should allow the user to post his score to facebook
  + The game should allow the user to tweet his score

Future Enhancements:

* Game Running on Mobile Devices
  + Target Resolution should be: 320x480

Collision Detection:

**Block detects when finger gesture or mouse clicks:**

The main interaction of the game is the player touching the board. The game should automatically detect when the player is clicking/touching a square, and handles destroying the square if it detects a match of 3 or more.

**Block detects when it is touching 3 or more objects:**

The block entity should be aware of when it is touching two matching squares. If a match is detected, the blocks should disappear and the player is awarded points.

**Block Detects When a block reaches the top of the screen:**

If a column of blocks reach the top of the screen, the game manager should run the **Lose Game** function. A failure sound effect should play, and the lose game function should call the Lose Game UI. (See UI section for details).



*Example of a player in danger as blocks are nearly touching the top of the screen*

Game Manager

The Game Manager is where most of the magic happens. The game manager controls the basic loop of game action. It handles User Interface, Generating the sprites/play field, managing the win/loss condition, and scoring.

Timer

The Timer should start at 1 minute. The timer constantly ticks down and displays the current time via the User Interface. If the timer reaches 0, and the player has not allowed any blocks to touch the top of the screen the Win Condition should be called.

**Level Loader**

The level loader is called when the timer ends. This function should handle the loading of the next level, and display the player’s score. The level loader should also update the speed variable of the row generator; making the level and game faster and more difficult.

**Score Manager**

The Score Manager handles the the events that occur when the player matches 3 or more blocks. The score manager calculates the appropriate score and sends it to the User Interface to communicate the score to the player. The score manager also handles the end of level scoring calculation.

The scoring is handled the following way:

Player is awarded 100 points for each match of 3. If the player gets more than 3 matching blocks, a multiplier is applied. The multiplier increases based on the number of blocks in the match. For example: A player who matches 3 blue blocks, he will get 100 points. If he matches 5 blue blocks, he gets 100 points plus a multiplier of 2 or 200 points.

**(Needs to be fleshed out more)**

**Win Condition**

The games should handle what happens when the timer runs down. This should work with the level loader, and GUI to display the final level score. On the next level, the rows appear faster.

**Loss Condition**

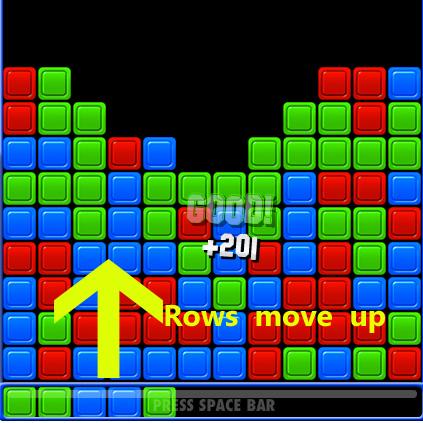
The game must have a loss condition. This should communicate with the UI to calculate and display the player’s final score. Any sprites or particles should be initiated here as well.

Sprite Manager/Play field Generator

The sprite manager/ playfield generator should handle placing objects in the world. The Row Generator is a child of this object. The sprite manager also handles the placement of particle effects on the screen. This pixel placement should be handled here.

**Row Generator**

The Row Generator runs as long as the game is in a Play State (The player hasn’t lost yet). The Row Generator should randomly create a row of 14 colors. When the row is complete, it pushes the row into the game space. As the player gets to the next level, the speed at which rows generate should increase. When the player clears matching blocks the remaining blocks should merge collapse in together. (For example, if the player clears blocks at the bottom, the blocks on top should fall.)



*Example of how rows move up in from the bottom*

Examples of how to clear blocks in this video: <https://www.youtube.com/watch?v=X97ANpVxOLY>

**User Interface**

The User Interface is one of the major ways that the game communicates with the player.

**Main Screen:**

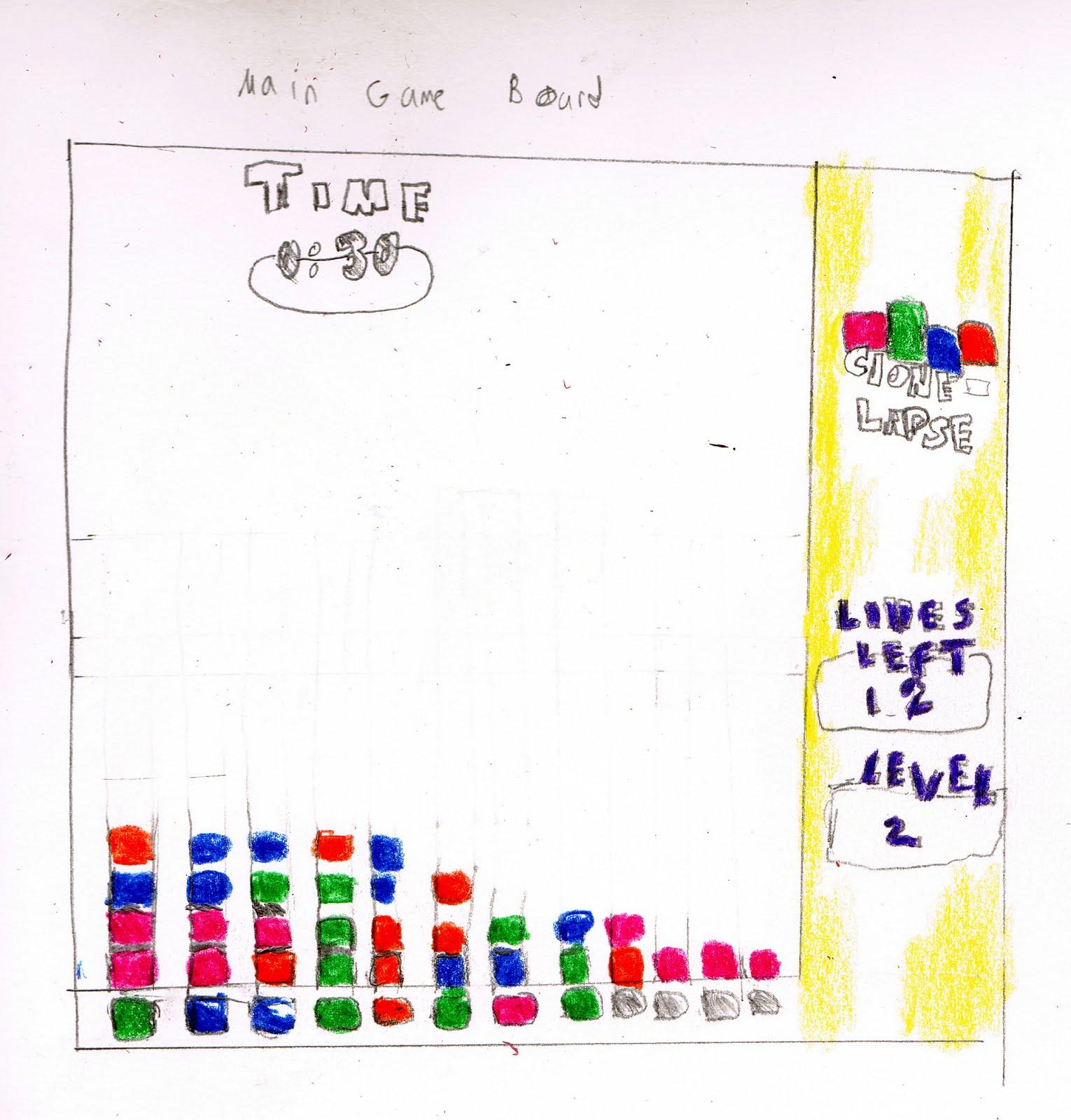
The Main Screen is the first thing that the player sees when loading the game. The user interface should give the user the following options.

* **Start Game:** This should load a brand new version of the level. Each time the player loads a new game, the block patterns should be randomized
* **Quit Game:** On Mobile only. This will not be in the web version, since there is no

way to quit the from the browser. This should exit the game, and return to the mobile device home screen.

**In Game UI:**

The In Game User Interface should handle the following:



* **Score**: The game manager piece handles collision and calculation of player points. The UI should handle the updating and displaying of this information
* **Levels**: The UI should constantly display the current level that the player is on. When the Level Loader is run at the start of each new level, the UI should update this field
* **Time**: The UI should constantly check and update the timer on screen.When time expires, this element should handle calling the ‘Win’ module and update the player’s score for that level.

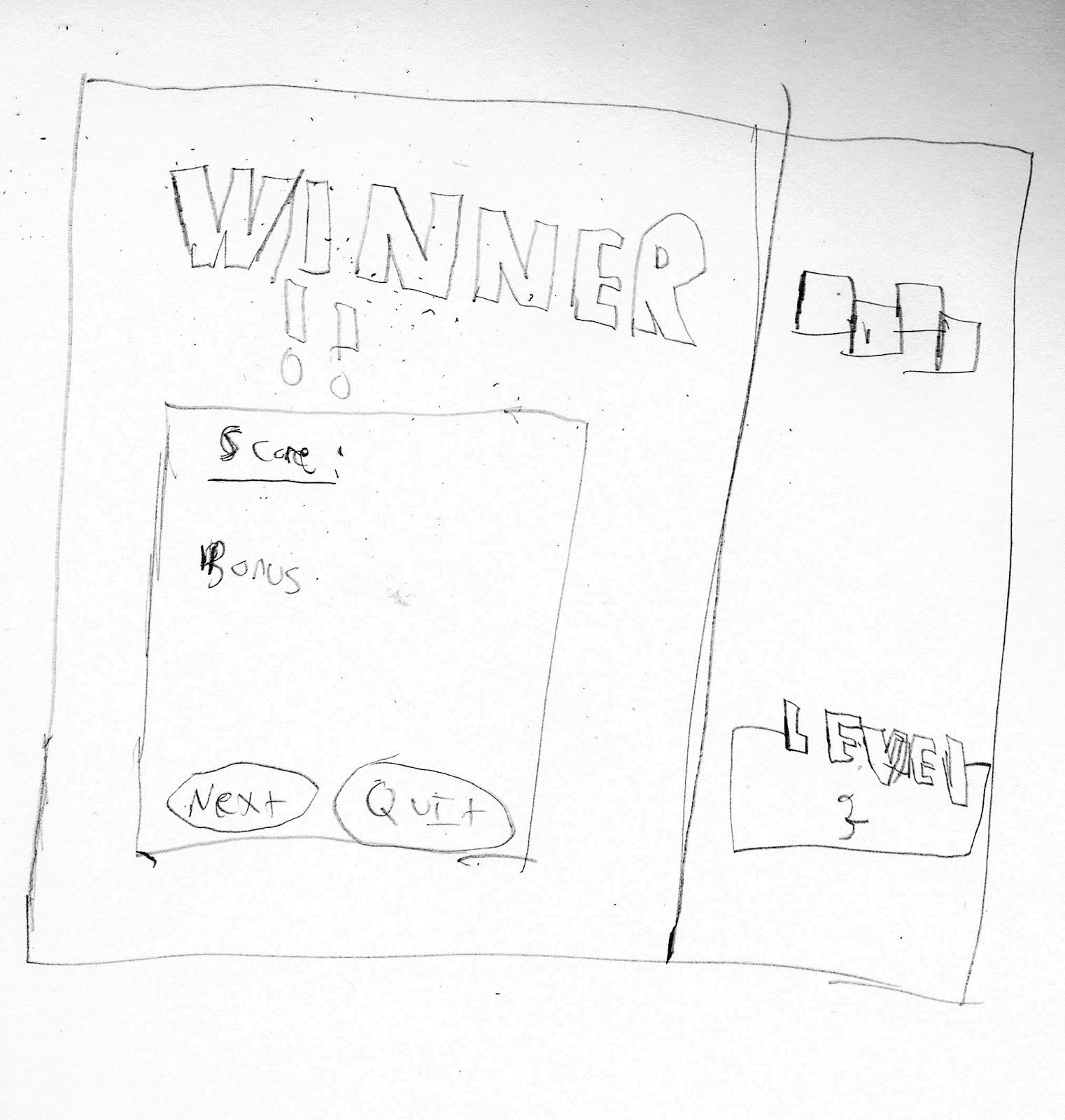
**Pause Menu UI:**

Should stop all action when hit. Should also display a UI asking if player wants to quit to main screen or continue.

**Win Screen:**

The Win Screen should pop up when the timer expires. When the timer expires victory music should play, particles should fly around the screen, and the victory text should fly in. This sequence will help the player feel like beating a level is an achievement. The Win Screen gives the player the following options.

* + **Quit:** Returns to the main menu
  + **Next:** Calls the level loader module and loads next level



*Win Screen Example*

**Loss Condition UI:**

When the player loses the game. The loss condition of the game manager is called. This UI shows the following:

* Play Again: This button should reload the current level
* Quit: This should return the player to the main screen

Game Running on Mobile Devices

The Game needs to be tuned to run on mobile devices. CSS should be used to detect the user’s mobile device size. For certain devices, we should lock in the target resolution if possible.

Non Functional Requirements

**Need a website to Host the game**

The website needs to host the canvas that allows players to experience the game.

The website should do the following:

* Detect where the user is coming from and resize appropriately
  + OS browser
    - Firefox
    - Chrome
    - IE9?
  + iPad
  + iPhone
  + Android phone
  + Windows Phone
* Feature a Database for storing the player’s high scores
  + The database needs a table that the game writes the highest scores to
  + Need to decide if this is stored globally, or locally

**Game Must have Sound**

The sound manager should handle both sound and music playing. The game should play a sound during the following times:

* Game Win
* Game Loss
* Blocks Matched/Destroyed
* Warning Sound when the blocks get near the top of the screen

The Game sound manager should detect the appropriate time to play the sounds above. They can also be baked in to the individual methods themselves.