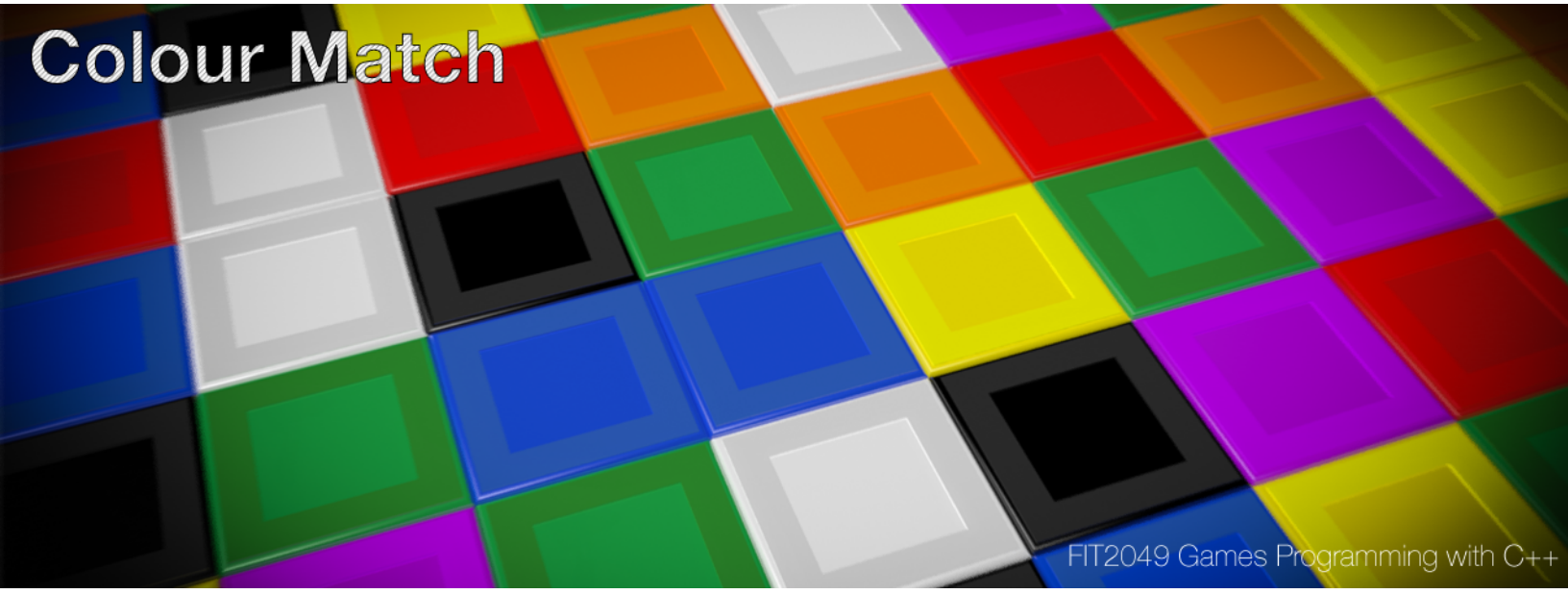
Simple Game Implementation

Due Date: Friday Week 5 (26th August, 11:55 PM) - **Weight:** 15%



The Task:

For your first assignment you'll be implementing a simple colour matching game inspired by the Bejeweled series and creatively titled "Colour Match". The game revolves around finding groups of the same colour on a 3D grid of cubes. The basic gameplay is as follows:

- When the game starts a grid of random colours will appear on screen.
- In a separate section of the screen, the player will be shown a random colour – this is known as the "Next Colour".
- Using the arrow keys, the player will select one of the cubes on the grid and when they press the spacebar key, the colour of the selected cube will change to match the "Next Colour".
- If the change in colour causes three or more cubes of the same colour to be connected, those cubes are removed from the screen and the player's score is increased (the amount of score increase is up to you, but they should get bigger scores for matching larger amounts of cubes). Cubes can only be connected by their four neighbours – diagonals don't count.
- After the matched cubes are removed, cubes above them on the grid should move down to fill their place (there should never be any holes in the board). New cubes should also appear along the top of the board to fill the place of any cubes which fell down to a lower row.
- After the cubes are removed, the "Next Colour" is randomised again and the player can pick another cube to replace.

When the game ends, a **game over screen** should be displayed which tells the player their final score. A quit and a restart button should also be present on this screen.

The end condition of the game is controlled by two things:

- If a move doesn't result in a match of three or more connected cubes, a life is lost. After losing three lives, the game ends.
- A move counter should also be included which limits the amount of allowed moves per game. The player should be able to see the amount of moves remaining somewhere in the UI.

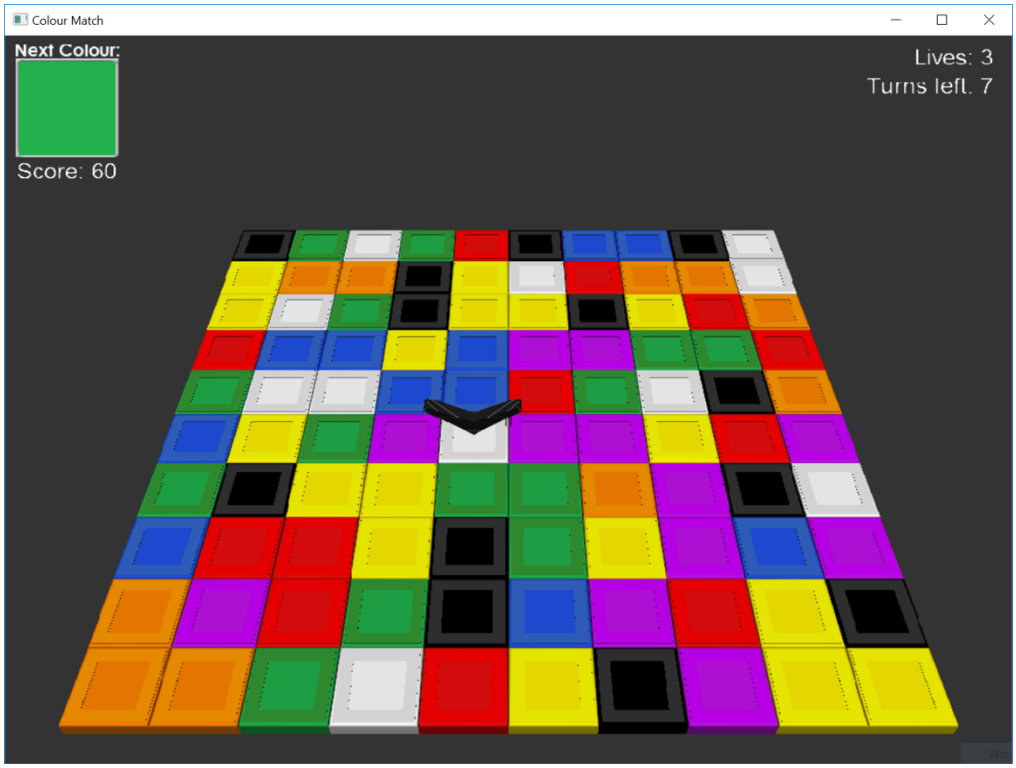
To spice up the gameplay, a special Power Cube should be included, as outlined below:

- When the player selects a cube on the grid and activates the power cube with the P key, every cube on the grid that matches the selected colour should be removed from the board.
- The score rewarded should be based on the amount of cubes that were removed.
- The power cube is available from the start of the game and can only be used once – use it wisely!

Provided Assets:

Provided inside the **Assignment 1 Assets** file on Moodle are all the assets you'll need to create the game. The cubes have a width and length of one unit allowing for easier placement. It is recommended you build your solution off the **Week 3 Base Code** (while waiting for that code to be released, you should spend week two thinking about your solution).

Your game might look something like this:



Please note that while additional assets may be included, they won't be considered when marking. Only your C++ code and the quality of your gameplay will be assessed (we're not marking your modelling skills).

Submission:

Your assignment must be uploaded to Moodle before the due date. Any late assignments will be penalised 5% of the total available mark per day late.

Students seeking extensions will have to follow the Special Consideration guidelines outlined here: http://www.monash.edu.au/exams/special-consideration.html (Note: you will need to provide adequate documentation to support your claim for an extension).

Please include an assignment coversheet with your submission. They can be found here: http://infotech.monash.edu.au/resources/student/forms/

There's a marking rubric on the next page – go check it out!

Marking Rubric:

Program Functionality – 40%					
	N	P	C	D	HD
Does the game initialise and use Direct 3D correctly?					
Is the game world drawn correctly?					
Is input handled correctly?					
Are all of the required assets loaded correctly?					
Are sprites and text displayed correctly?					
Game Functionality – 40%					
	N	P	C	D	HD
Is the selection arrow implemented correctly?					
Are matching cubes correctly identified?					
Are the cubes removed in an interesting and convincing way?					
Are the game rules implemented well?					
What is the quality of the game over screen?					
Code Quality – 20%					
	N	P	C	D	HD
Is the code written in an efficient manner? (including good memory management)					
Does the code exhibit good object-oriented design?					
Is the readability and style of the code to a high standard? Are **appropriate comments** included throughout?					